

SAC Manual and Tutorial

Brian Savage, Peter Goldstein, and Arthur Snoke

Contents

1	Tutorial	1
2	SAC User Manual	10
2.1	Introduction	10
2.2	New Users	13
2.3	Analysis Capabilities	16
2.4	Graphics Capabilities	22
2.5	SAC Macors	25
2.6	Inline Functions	34
2.7	Blackboard Variables	40
2.8	Using SACIO	43
2.8.1	SAC Data File Format	52
2.8.2	API	64
2.8.3	API Howto	69
2.8.4	Subroutine Calls	69
3	Commands	78
3.1	EXM: Executive Module	78
3.1.1	About	78
3.1.2	Cd	78
3.1.3	Comcor	79
3.1.4	Done	79
3.1.5	Echo	79
3.1.6	Evaluate	80
3.1.7	Funcgen	82
3.1.8	Getbb	84
3.1.9	Help	86
3.1.10	Inicm	87

3.1.11	Installmacro	87
3.1.12	Load	88
3.1.13	Macro	89
3.1.14	Message	90
3.1.15	News	90
3.1.16	Pause	91
3.1.17	Printhelp	91
3.1.18	Production	92
3.1.19	Readbbf	92
3.1.20	Report	93
3.1.21	Setbb	94
3.1.22	Setmacro	95
3.1.23	Syntax	96
3.1.24	Systemcommand	96
3.1.25	Trace	97
3.1.26	Transcript	98
3.1.27	Unsetbb	100
3.1.28	Writebbf	101
3.2	DFM: Data File Module	101
3.2.1	Chnhdr	101
3.2.2	Convert	104
3.2.3	Copyhdr	104
3.2.4	Cut	105
3.2.5	Cuterr	108
3.2.6	Cutim	109
3.2.7	Datagen	111
3.2.8	Deletechannel	115
3.2.9	Headerwindow	115
3.2.10	Listhdr	116
3.2.11	Pickauthor	117
3.2.12	Pickphase	118
3.2.13	Pickprefs	119
3.2.14	Read	122

3.2.15	Readalpha	129
3.2.16	Readcss	129
3.2.17	Readerr	135
3.2.18	Readgse	136
3.2.19	Readhdr	138
3.2.20	Readsdd	140
3.2.21	Readsuds	142
3.2.22	Sort	146
3.2.23	Synchronize	147
3.2.24	Wild	149
3.2.25	Write	152
3.2.26	Writecss	156
3.2.27	Writegse	157
3.2.28	Writehdr	159
3.2.29	Writesdd	160
3.2.30	Writetable	161
3.3	GCM: Graphics Control Module	161
3.3.1	Begindevices	161
3.3.2	Enddevices	163
3.3.3	Erase	163
3.3.4	Hcd	163
3.3.5	Vspace	164
3.4	GEM: Graphic Environment Module	165
3.4.1	Axes	165
3.4.2	Beginframe	166
3.4.3	Beginwindow	168
3.4.4	Border	169
3.4.5	Color	169
3.4.6	Colortable	171
3.4.7	Endframe	171
3.4.8	Floor	171
3.4.9	Grid	172
3.4.10	Gtext	172

3.4.11	Line	174
3.4.12	Linlin	175
3.4.13	Linlog	175
3.4.14	Loglab	176
3.4.15	Loglin	176
3.4.16	Loglog	176
3.4.17	Null	176
3.4.18	Plabel	177
3.4.19	Qdp	178
3.4.20	Symbol	180
3.4.21	Ticks	182
3.4.22	Title	183
3.4.23	Tsize	184
3.4.24	Wait	185
3.4.25	Width	186
3.4.26	Window	188
3.4.27	Xdiv	189
3.4.28	Xfudge	189
3.4.29	Xfull	190
3.4.30	Xgrid	191
3.4.31	Xlabel	191
3.4.32	Xlin	192
3.4.33	Xlog	192
3.4.34	Xvport	192
3.4.35	Ydiv	193
3.4.36	Yfudge	194
3.4.37	Yfull	195
3.4.38	Ygrid	195
3.4.39	Ylabel	195
3.4.40	Ylin	196
3.4.41	Ylog	196
3.4.42	Yvport	197
3.5	GAM: Graphic Action Module	198

3.5.1	Fileid	198
3.5.2	Filename	199
3.5.3	Picks	199
3.5.4	Plot	201
3.5.5	Plot1	201
3.5.6	Plot2	203
3.5.7	Plotalpha	204
3.5.8	Plotc	207
3.5.9	Plotdy	211
3.5.10	Plotpk	212
3.5.11	Plotpm	214
3.5.12	Plotxy	215
3.5.13	Print	216
3.5.14	Setdevice	217
3.5.15	Xlim	217
3.5.16	Ylim	218
3.6	SAM: Spectral Analysis Module	219
3.6.1	Bandpass	219
3.6.2	Bandrej	222
3.6.3	Benioff	223
3.6.4	Convolve	224
3.6.5	Correlate	225
3.6.6	Dft	226
3.6.7	Divomega	226
3.6.8	Envelope	227
3.6.9	Filterdesign	228
3.6.10	Fir	229
3.6.11	Hanning	230
3.6.12	Highpass	231
3.6.13	Hilbert	232
3.6.14	Idft	232
3.6.15	Keepam	232
3.6.16	Khronhite	233

3.6.17	Lowpass	233
3.6.18	Mulomega	234
3.6.19	Plotsp	234
3.6.20	Readsp	236
3.6.21	Unwrap	237
3.6.22	Wiener	238
3.6.23	Writesp	240
3.7	UOM: Unary Operations Module	242
3.7.1	Abs	242
3.7.2	Add	242
3.7.3	Dif	243
3.7.4	Div	243
3.7.5	Exp	244
3.7.6	Exp10	244
3.7.7	Int	245
3.7.8	Log	245
3.7.9	Log10	245
3.7.10	Mul	246
3.7.11	Sqr	246
3.7.12	Sqrt	247
3.7.13	Sub	247
3.8	BOM: Binary Operations Module	248
3.8.1	Addf	248
3.8.2	Binoperr	249
3.8.3	Divf	250
3.8.4	Merge	251
3.8.5	Mulf	253
3.8.6	Subf	254
3.9	EAM: Event Analysis Module	255
3.9.1	Apk	255
3.9.2	Ccipf	257
3.9.3	Chpf	257
3.9.4	Ocipf	258

3.9.5	Ohpf	258
3.9.6	Whpf	259
3.10	SCM: Signal Correction Module	259
3.10.1	Decimate	259
3.10.2	Interpolate	260
3.10.3	Linefit	261
3.10.4	Quantize	262
3.10.5	Reverse	263
3.10.6	Rglitches	263
3.10.7	Rir	265
3.10.8	Rmean	265
3.10.9	Rotate	265
3.10.10	Rq	267
3.10.11	Rtrend	268
3.10.12	Smooth	268
3.10.13	Stretch	269
3.10.14	Taper	269
3.11	SPE: Spectral Estimation Subprocess	270
3.11.1	Cor	270
3.11.2	Mem	270
3.11.3	Mlm	271
3.11.4	Pds	271
3.11.5	Plotcor	271
3.11.6	Plotpe	271
3.11.7	Plotspe	271
3.11.8	Prewhiten	271
3.11.9	Quitsub	271
3.11.10	Read	271
3.11.11	Readcor	278
3.11.12	Spe	279
3.11.13	Speid	283
3.11.14	Writecor	283
3.11.15	Writespe	283

3.12	SSS: Signal Stacking Subprocess	283
3.12.1	Addstack	283
3.12.2	Changestack	283
3.12.3	Deletestack	283
3.12.4	Deltacheck	283
3.12.5	Distanceaxis	283
3.12.6	Distancewindow	283
3.12.7	Globalstack	283
3.12.8	Incrementstack	283
3.12.9	Liststack	283
3.12.10	Phase	283
3.12.11	Plotrecordsection	284
3.12.12	Plotstack	284
3.12.13	Quitstack	284
3.12.14	Sss	284
3.12.15	Sumstack	286
3.12.16	Timeaxis	286
3.12.17	Timewindow	286
3.12.18	Traveltime	286
3.12.19	Velocitymodel	286
3.12.20	Velocityrosette	286
3.12.21	Writestack	286
3.12.22	Zerostack	286
3.13	SMM: Signal Measurement Module	287
3.13.1	Markptp	287
3.13.2	Marktimes	288
3.13.3	Markvalue	289
3.13.4	Mtw	290
3.13.5	Rms	291
3.14	ICM: Instrument Correction Module	292
3.14.1	Prewhten	292
3.14.2	Transfer	292
3.15	CND: Conditional Execution Module	300

3.15.1	Break	300
3.15.2	Do	300
3.15.3	Else	300
3.15.4	Elseif	300
3.15.5	Enddo	300
3.15.6	Endif	300
3.15.7	If	301
3.15.8	While	301
3.16	NNM: Neural Network Module	301
3.16.1	Writenn	301
3.17	XYZ: XYZ (3-d) Data Processing Module	301
3.17.1	Contour	301
3.17.2	Grayscale	303
3.17.3	Image	305
3.17.4	Sonogram	306
3.17.5	Spectrogram	307
3.17.6	Zcolors	310
3.17.7	Zlabels	310
3.17.8	Zlevels	311
3.17.9	Zlines	312
3.17.10	Zticks	313
3.18	FKS: frequency-wavenumber (k) spectral analysis	314
3.18.1	Arraymap	314
3.18.2	Bb fk	314
3.18.3	Beamform	317
3.18.4	Gmtmap	317
3.19	MAT: matlab analysis routines	317
3.19.1	Closemat	317
3.19.2	Mat	317
3.19.3	Mat3c	319
3.19.4	Matdepmec	319
3.19.5	Recordsection	319
3.19.6	Setmat	320

3.20 CODA: Kevin Mayeda's coda magnitude	320
3.20.1 M coda	320
4 SAC Graphics File (SGF)	320
5 History	323
6 FAQ	333
7 Availability	339
8 Errors/Messages	340
9 Developers	348
10 Copyright	348
11 Privacy and Legal Notice	348

1 Tutorial

SAC2000 User's Manual

Tutorial

Purpose SAC2000 is a general purpose interactive program designed for the study of sequential data, especially time-series data. Emphasis has been placed on analysis tools needed by research seismologists in the detailed study of seismic events.

Index Differences Between Versions Getting Started SAC Data Files Reading and Writing Data Files Seeing the Results What Else Can I Do?

Overview This guide shows you by example how the basic SAC commands work. Trying these examples as you read the guide will quicken the learning process. If you want to learn more about a particular command, see the Command Reference Manual. For general information on how SAC2000 works, how to create and use SAC macros, on the structure of SAC data

files, and how to interface other programs to SAC2000, see earlier sections of this User's Manual.

Update Policy This guide will be periodically updated to include new information and to revise old information. It is suggested that it be kept in a loose leaf binder, to make it easier to incorporate these updates. The pages are not numbered for this reason. Please report any errors in this guide to the author. This will help keep it as accurate and current as possible.

If You Need Help Ask a knowledgeable friend! This often works best. If you have further questions, suggestions, or gripes of any kind feel free to contact one of us. We will always listen and can often help.

Peter Goldstein Mail Stop L-205 Lawrence Livermore National Laboratory Livermore, CA 94550 Email: peterg@llnl.gov

A Final Word SAC2000 is a large program with many capabilities and options. It can be confusing at first. Don't despair. The most important commands are discussed in this guide. You need to learn about the rest of the commands only as you need them. Common sense defaults exist for most options. SAC2000 does a lot of error checking so you can't get into too much trouble. Good luck and have fun!

SAC2000 Tutorial Guide for New Users

Differences Between Versions Each Computer Is Different

SAC2000 runs on several different kinds of UNIX computers: Sun/Solaris (this is the development platform and is always the most up-to-date) Sun/SunOS v4.x SGI/IRIX v6.2 PC/Linux 2.0.30 Alpha/DEC OSF/1 V3.2 (Rev. 214) (this is brand new as of 10/97. Consider it as a beta version)

SAC2000 works similarly but not identically on each of these computers. This section includes instructions to help you set up your environment to be able to run SAC2000 on each of these types of computers. The instructions are short and easy to implement. Once you have completed them, you will be ready to start using SAC2000. If they are not clear or if you have problems please see your system administrator or the person who installed SAC2000 on your system.

First set up the environmental variable SACAUX to be the name of the directory that contains the SAC auxiliary files. These files are used by

SAC2000 while it is executing. Assume that this directory is located on your system in the directory /foo/sac/aux. If you are running under AT&T System V you would put the following in your .profile file:

```
SACAUX = /foo/sac/aux export SACAUX
```

If you are running under BSD 4.2 you would put the following in your .login file:

```
setenv SACAUX /foo/sac/aux
```

It is important that SACAUX be in uppercase and that the pathname be entered in the correct case. The SAC2000 executable is called sac2000 and is normally found in the bin subdirectory (i.e. /foo/sac/bin/sac). Add this subdirectory to your search path or set up an alias. Once you have completed these two steps, to start up the program type “sac2000”.

Getting Started

Starting up SAC2000 Once you have set up your environment as discussed in the last section and are logged onto a graphics terminal or workstation, simply type “sac2000”. SAC2000 will then print a short headline including the number and date of the version you have on your system. It may also print a bulletin giving some current information. SAC2000 will then ask you for input by sending the prompt “SAC”.

Interaction SAC2000 is an interactive command driven program. This means that you must type a command to get SAC2000 to do something. It does not prompt you for input. Commands may be typed at the terminal or placed in a command file. Symbols within a command are separated by spaces and commands within a given line may be separated by a semicolon.

A Simple Example We'll start by creating a simple function:

```
FUNCGEN IMPULSE
```

This generates an impulse function and stores it in SAC2000's memory. To see what this function looks like on your screen type:

```
BEGINDEVICES device PLOT
```

In this example device is the name of the graphics device you are using. If you don't use the BEGINDEVICES command, SAC2000 will use the default device which is X windows on most.

Abbreviations There are abbreviations for the most used SAC com-

mands. For example, FG, BD, and P are the abbreviations for FUNCGEN, BEGINDEVICE, and PLOT respectively. Most options also have abbreviations: X for XWINDOWS. (There is a graphics device for generating hardcopy plots. It is called SGF for SAC Graphics File. There is also SUNWINDOWS for those using SunView.

More functions The FUNCGEN command can generate a number of different functions. This is very useful when first learning how to use SAC2000 because you can see how the other SAC operations work on these functions. For example, type:

```
FUNCGEN SEISMOGRAM
```

This generates a sample seismic signal in SAC2000's memory. It also deletes the impulse generated earlier. Use the PLOT command to see this seismogram on your screen. Now for another function:

```
FUNCGEN SINE 2 NPTS 200 DELTA 0.01
```

This is an example of a more complicated SAC command. This example generates a 2 Hz sine wave in SAC2000's memory. The function will contain 200 data points and have a sampling interval of 0.01 seconds. You may want to use the PLOT command to plot this function also.

SAC Commands There are several general points to be made at this point about SAC commands. All input is space delimited. The decimal point is optional wherever numeric input is needed. When you specify a value for a particular option, this value becomes the new current value. This means you don't have to keep entering values for options that you don't want to change. For example, you can now generate this same 2 Hz sine wave using the same sampling interval but with 400 data points by simply typing:

```
FUNCGEN NPTS 400
```

SAC commands fall into two main categories: parameter-setting and action-producing. The parameter-setting commands basically change values of internal SAC parameters. Action-producing commands perform some operation on the data files currently in memory based upon the values of these same parameters. The effect of a parameter-setting command remains in effect until it is reset. The effect of an action-producing command, however, is immediate and transitory. For example, the parameter-setting command,

YLOG, tells SAC2000 to use logarithmic interpolation for the y axis in subsequent plots. The action-producing command, PLOT, does the actual plotting. Options to action-producing commands also remain in effect until reset just like parameter-setting commands. The underlying assumption is that you are more likely than not to want to use the same values the next time you execute the same command.

Default Values All commands have “nice” default values for most options. The use of current and default values for command options can save you a lot of typing. For example, let’s look at the BANDPASS command. This command applies a bandpass filter to the data currently in memory:

```
FUNCGEN IMPULSE NPTS 100 DELTA .01 BANDPASS BESSEL  
CORNER .1 .3 NPOLES 4
```

These two commands generate an impulse function and then apply a bandpass filter to that impulse. The filter is a four-pole Bessel filter with corner frequencies at 0.1 and 0.3 Hz. You can see the result in the time domain by typing PLOT or you can see the amplitude response by taking the Fourier transform and using the PLOTSP command:

```
FFT PLOTSP AM
```

You can now try a different set of corner frequencies very easily:

```
FUNCGEN BANDPASS CORNER .2 .5
```

SAC2000 generates the same impulse function and applies the same Bessel filter except for the new corner frequencies.

SAC Data Files

What is a SAC Data File?

SAC2000 is a program to examine, analyze, and plot data. This data is stored on disk as SAC data files. Each data file contains a single data set. For seismic data this means a single data component recorded at a single seismic station. SAC2000 does not currently work on multiplexed data. The data will generally be evenly spaced time series data. SAC2000 can also handle unevenly spaced data and spectral data. The spectral data can be in either real-imaginary or amplitude-phase format.

The SAC Header Each data file also contains a header record which describes the contents of that file. Certain header entries are always present

(e.g., the number of data points, the file type.) Others are always present for certain file types (e.g., sampling interval, begin time, etc. for evenly spaced time series files.) Other header variables provide information needed by a particular operation (e.g., seismic component orientation used by the ROTATE command.) Still others are not used by SAC2000 at all. They are simply informational. SAC data files, including all header information, can be read and written by user created programs as well as by SAC2000. See the SAC2000 User's Manual for details. The LISTHDR command displays the contents of the headers for the data files currently in memory. You may wish to examine the header from the sample seismogram mentioned earlier:

```
FUNCGEN SEISMOGRAM LISTHDR
```

If a particular header variable does not have a value for a particular file, then that variable is said to be “undefined” for that file. The LISTHDR command does not list undefined header variables, unless it is invoked with the INC or INCLUSIVE option (which includes undefined header variables).

Header Variables Each header variable is described in the Users Manual. The most important ones are also listed below:

- NPTS Number of points in data set.
- B Beginning value of the independent variable.
- E Ending value of the independent variable.
- IFTYPE Type of file.
- LEVEN TRUE if data set is evenly spaced.
- DELTA Increment between evenly spaced samples.
- IDEP Type of dependent variable.
- KZDATE Alphanumeric form of GMT reference date.
- KZTIME Alphanumeric form of GMT reference time.
- A First arrival time (seconds relative to reference time.)
- T n User defined time picks or markers, n=0,9.

Reading and Writing Data Files

The READ Command SAC commands work on data already in SAC2000's working memory, not data on disk. The READ command is used to transfer data from disk to memory. Up to 200 data files can be in memory at the same time. These can be of any size up the maximum size of SAC2000's

working memory. This value is normally 1,000,000 32-bit words. (Both the number of files and the size of SAC2000's working memory can be changed by making minor modifications to the source code and recompiling the program.) You can use wildcard characters in the READ command to represent groups of files which have a similar set of characters in their names. Each time you use the READ command to transfer data from disk to memory the data currently in memory is destroyed. If you want this data saved, you must write it to disk before reading more data into memory. There is an option called MORE in the READ command that lets you read data into memory without destroying the old data. See the Command Reference Manual for details.

The WRITE Command Action commands (such as ADD, DECIMATE, and FFT) modify the data that is currently in memory. The data files on disk are not modified. At any time during your analysis, you may transfer this modified data back to disk using the WRITE command. You may overwrite the old data files on disk using the OVER option or create new ones by specifying their file names.

Several Examples The examples below demonstrates several uses of the READ and WRITE commands. The first example reads two files into memory, multiplies each data point in each file by a constant, and then writes the results to disk in two new files:

```
READ FILE1 FILE2 MUL 10 20 WRITE FILE3 FILE4
```

The next example reads a single file into memory, desamples the data by a factor of five (DECIMATE also applies an anti-aliasing filter), and then writes the results back to disk using the same file name:

```
READ FILE5 DECIMATE 5 WRITE OVER
```

Sample Data Files You're going to need some data files for use in the next section on plotting. You'll also need them if you want to try any of the other commands discussed later in this guide. If you don't have any sample SAC data files around to play with, you can use FUNCGEN to generate some. This is shown in the example below:

```
FUNCGEN TRIANGLE NPTS 200 DELTA 1 WRITE FILE1 FUNC-  
GEN BOXCAR WRITE FILE2 FUNCGEN STEP WRITE FILE3
```


This results in you having three files in your directory called FILE1, FILE2, FILE3 which contain the triangle and boxcar, and step functions respectively. Each will have 200 data points in them and be sampled at 1 sample per second. If you already had files in your directory by those names, they would be replaced by these new ones.

Some Real Seismic Data If you want to use some real seismic data files instead of the simple functions generated above, you can use the DATAGEN command. It has three-component data from three different seismic events (a local, a regional, and a teleseism) that were recorded at Livermore. For example, to generate three different vertical components from the local event you could type:

```
DATAGEN SUB LOCAL CAL.Z CAO.Z CDA.Z WRITE FILE1 FILE2  
FILE3
```

Or if you wanted all three components from a single station in the teleseismic event you could type:

```
DATAGEN SUB TELESEIS NYKL.Z NYKL.N NYKL.E WRITE FILE1  
FILE2 FILE3
```

Each of these events, including some information about the recording network, the length of the files, and the file names can be found in the DATAGEN documentation in the Command Reference Manual.

Seeing the Results Overview After reading data into SAC2000 you can see it on your screen in several different formats using the various plot commands. Default values for each of the graphics display commands have been chosen to make it as easy as possible to display your data. By changing these default values before plotting, you also have complete control over the details of how each plot will look.

PLOT You've already used PLOT to display data files. With this command, each data file is plotted one at a time. SAC2000 pauses between files to give you a chance to examine the data. This is shown in the following example. User responses (what you type) are preceded by a "u:" and SAC2000's responses by an "s:".

```
u: READ FILE1 FILE2 FILE3 read in 3 files u: PLOT SAC2000 plots  
first file to your terminal. s: Waiting after looking at plot type a carriage
```

return.} u: [return] SAC2000 plots second file. s: Waiting look at second plot. u: [return] SAC2000 does not pause after the last plot.

There are other responses to this prompt that allow you to cancel the remainder of the plots or see them without pausing.

Other Plot Commands Several other canned plot formats are available. PLOT1 plots each file along a common x axis but with a separate y axes. By default all files are placed on the same plot. Try this with the three files from the example above. PLOT2 is an overlay plot. Again all files are plotted together, this time using both a common x and a common y axis. PLOTPK uses a format similiar to PLOT1. It lets you use the cursor to blow up parts of the plot, determine values of selected data points, pick phase arrival times, etc.

Plotting Options By default, all SAC2000 plots are self-scaling. SAC2000 determines what limits to use for the x and y axes. If you want to set these limits yourself, you may do so using the XLIM and YLIM commands. If you wish, you may also change the location of annotated axes, change the linestyle, select a symbol to be plotted at each data point, create titles and labels, make logarithmic plots, change the size and type of text, and control a number of other even more exotic aspects of the plot. These commands are part of the Graphics Environment Module. They are described in the Users Manual and explained in detail in the Command Reference Manual.

What Else Can I Do? Overview Fortunately (for you and for me) SAC2000 does a lot more than just reading, writing, and plotting data files! Some of SAC2000's analysis capabilities are briefly discussed below.

Filtering FFT and IFFT take the Fourier and inverse Fourier transform time series data. LOWPASS, HIGHPASS, BANDPASS, and BANDREJ are a set of Infinite Impulse Response (IIR) filters. You may choose from Butterworth, Bessel, and Chebyshev Type I and II filters. WIENER applies an adaptive Wiener filter. FIR applies a Finite Impulse Response filter. DECIMATE applies an anti-aliasing lowpass filter as it desamples data. UNWRAP computes a spectral amplitude and an unwrapped phase.

Unary and Binary Operations You may add a constant to each data point in a file using the ADD commands This is called a unary operation in

SAC2000. Other unary commands include SUB, MUL, DIV, SQR, SQRT, EXP, and LOG. You may also add two data files together using the ADDF command. This is called a binary operation in SAC2000. Other binary commands include SUBF, MULF, DIVF, and MERGE.

Correcting Signals There are a number of commands available to correct or modify seismic signals. RQ removes the seismic Q factor from spectral data. RTM, RTREND, and RMEAN remove timing marks, the linear trend, and the mean, respectively, from time series data. TAPER applies a symmetric taper to each end of the data. ROTATE rotates a pair of data components through an specified angle in the plane of the components.

Phase Picking APK applies an automatic event picking algorithm to seismic data. Output can be written to a HYPO formatted disk file or to a more general alphanumeric pick file. PLOTPK can also be used to pick and write phase information into these files.

Summary This is only a partial list of SAC2000's analysis capabilities. The list grows with each release. A much more complete list is given in the Users Manual. Each command is described fully in the Command Reference Manual. If you have some ideas on commands or features that you think are missing, let me know. They just might wind up in the next version.

2 SAC User Manual

2.1 Introduction

SAC2000 User's Manual

Introduction

Introduction SAC2000 (Seismic Analysis Code for the third millenium) is a general purpose interactive program designed for the study of sequential signals, especially time-series data. Emphasis has been placed on analysis tools used by research seismologists in the detailed study of seismic events. Analysis capabilities include general arithmetic operations, Fourier transforms, three spectral estimation techniques, IIR and FIR filtering, signal stacking, decimation, interpolation, correlation, and seismic phase picking.

SAC2000 also contains an extensive graphics capability. Versions are available for a wide variety of computer systems. SAC2000 was developed at Lawrence Livermore National Laboratory and is copyrighted by the University of California.

This manual

This manual contains general information for the new user about what SAC2000 can do, how it works, and how to get started. It also contains detailed information for the more experienced user on topics such as how to use SAC macros, how to read and write SAC data files from C or FORTRAN programs, and how the SAC2000 program is structured.

Table of Contents:

Introduction Provides an introduction, a table of contents, an update policy, and discussions of notation, other SAC manuals, design philosophy, and version differences.

For New Users An overview of SAC2000 and some suggestions on getting started.

Analysis Capabilities A brief description of each of the analysis commands.

Graphical Capabilities A brief description of the graphics devices and each of the graphics commands.

SAC Macros A detailed description on how to create and use SAC macro files to repeatedly execute a group of SAC commands.

Inline Functions A detailed description of a set of inline functions for performing numeric calculations and character string manipulations within other commands.

Blackboard Variables The blackboard is a feature that can be used to temporarily store and retrieve information while in SAC2000.

Reading And Writing SAC Data Files In Home Grown Software A description of how to read and write SAC data files in your own C or FORTRAN program. Also discussed is how to access specific header variables.

SAC Data File Format A detailed description of the SAC data file format, including the header.

Appendix: Subroutine Calls A description, including the calling sequence, of each of the subroutines discussed in this manual.

Update Policy

This manual will be periodically updated to include new descriptions and to revise old ones. It is suggested that it be kept in a loose leaf binder, to make it easier to incorporate these updates. The pages are not numbered for this reason. Please report any errors in this manual to: Peter Goldstein Mail Stop L-205 Lawrence Livermore National Laboratory Livermore, CA 94550 Email: peterg@llnl.gov

This will help keep it as accurate and current as possible.

Notation

This section describes the notation used in this manual. All of the SAC2000 manuals use a similar notation.

Uppercase words (e.g. READ) identify commands or keywords. They must be entered as shown, although they may be either uppercase or lowercase.

When showing examples of interaction between the user and SAC2000, the user inputs are denoted by “u:”, and SAC2000’s responses by “s:”. A uppercase typewriter style font is used within these examples, with comments about what is happening appearing in lowercase and enclosed in parentheses.

Repeating an important point made above, you may enter keywords and options in either uppercase or lowercase. SAC2000 converts these to uppercase before interpreting them. The exceptions to this rule are text appearing within single or double quotes and the names of directories and files. The case of these items is not changed. They are interpreted literally.

Other manuals Other SAC2000 manuals include:

A Tutorial Guide For New Users which explains the basic SAC commands with examples for you to try as you read.

Commands Reference Manual which contains detailed descriptions of each SAC command including purpose, syntax, default values, and examples. This manual also contains lists of SAC commands sorted alphabetically and functionally.

Spectral Estimation Subprocess Manual which describes a subprocess

for the study of stationary random processes. A subprocess is like a small separate program within the main SAC2000 program.

Signal Stacking Subprocess Manual which describes a subprocess for performing signal stacking with delays, traveltimes, and record section plots.

SAC Graphics File Users Manual which describes a set of programs that can be used to perform various functions on SAC Graphics Files.

Program Design

Design Philosophy

SAC2000 does not use a channel or a stack design. There are advantages and disadvantages in each of these designs. Concurrent operation was chosen because seismologists tend to perform the same operation on large numbers of seismic signals at the same time. This design sometimes requires you to store intermediate analysis results in temporary scratch files on disk. SAC2000's design was modified in version 10.6e to allow operations on subsets of files in memory to help minimize the need for writing temporary disk files.

Version Differences SAC2000 is a fairly portable code written in C. Two areas that effect portability are operating systems and graphics. There are currently versions for the following different types of computers and UNIX operating systems:

SUN: compiled on Solaris 5.5 and SunOS 4.1.1. Solaris is the platform on which SAC2000 is developed, it is generally the most up-to-date version.

SGI: IRIX 6.2

PC: Linux 2.0.30

DEC Alpha: OSF/1 V3.2 (Rev. 214) (this is brand new as of 10/97. Consider it as a beta version)

Old FORTRAN versions of SAC have also been ported to HP and IBM RS6000. SAC2000 works similiarly but not identically on each of these computers.

SAC2000 handles the second area, graphics, by including three different graphics devices in the code. They are:

XWINDOWS:

A windowing scheme that is available on most graphics workstations.

SUNWINDOW:

A windowing scheme that is available on SunOS v4.X.

SGF:

Stands for SAC Graphics File. Each file contains all the information needed to generate a single plot.

Each of these devices is described more fully in a later section of this manual. Check with your system administrator, if you need help.

2.2 New Users

SAC2000 User's Manual

For New Users

Overview SAC2000 was designed as an aid to research seismologists in the study of seismic events. As such, it is used for quick preliminary analyses, for routine processing, for testing new techniques, for detailed research, and for creating publication quality graphics. It is used by both computer novices and experts. In order to make SAC2000 quick to learn and easy to use, default values for all operational parameters were carefully chosen. At the same time, almost all of these parameters are under direct user control. This design combines ease of use with significant flexibility.

User Interface SAC2000 is an interactive command driven program. Commands may be typed at the terminal or placed in a macro file. SAC commands fall into three main categories: parameter-setting, action-producing and data-set manipulation. The parameter-setting commands change values of internal SAC parameters. Action-producing commands perform some operation on the signals currently in selected memory based upon the values of these parameters. Data-set commands determine which files are in active (selected) memory and therefore will be acted upon (data-set commands are not currently operational). The effect of a parameter-setting command remains in effect until it is reset. The effect of an action-producing command is immediate and transitory. Action-producing commands also have options which normally remain in effect until reset. These options, however, apply only to that particular command. The underlying assumption is that you

are more likely than not to want to use the same values the next time you execute the same command. When you start up SAC2000, default values are defined for all of these parameters. SAC2000 can be reinitialized to this default state at any time by executing the INICM command.

Mode of Operation Each signal is stored in a separate data file. Each data file contains a header that describes the contents of that file. See the section on Data File Format for details. Signals are read from disk into memory using the READ command. CSS 3.0 formatted flat files can be read using the READCSS command. SAC2000 can process up to 200 signals of arbitrary size at a time. Once data is in memory other commands are typed at the terminal (or read from a macro file) to perform operations on these signals. All operations work concurrently on ALL signals in memory. You can look at the results at any time using the plot commands. There are several plot formats to choose from. You have control over titles and labels, plot limits, file identifications, axes and tick mark locations, etc. You can also save the results of these operations at any time using the WRITE command. All of the commands are described briefly in the sections on Analysis Capabilities and Graphics Capabilities of this manual and documented in detail in the Commands Reference Manual.

How SAC2000 Handles Time The SAC header contains a reference or zero time, stored as six integers (NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, NZMSEC), but normally printed in an equivalent alphanumeric format (KZDATE and KZTIME.) This can be set to any reference time you wish. It is often the time of the first data point, but can also be the origin time of the event, midnight, your birthday, etc. It does not even have to be a time encompassed by the data itself. All other times are offsets in seconds from this reference time and are stored as floating point values in the header:

B = Begin time of the file. E = End time of the file. O = Event origin time. A = First arrival time. F = Fini (end of signal.) Tn = Time markers, where n is an integer from 0 to 9.

Many SAC commands work with these header variables. They are discussed in the two sections on Analysis and Graphics Capabilities. The CUT command is one of the most important of these commands. It lets you se-

lect portions of data files using the offset time header fields to be read in by subsequent READ commands. Examples of the use of CUT and READ are shown in the Tutorial Guide and also in the Command Reference Manual.

Getting Started The second best way to get started (after you have finished reading this section) is to sit down at a terminal or workstation with a copy of the Tutorial Guide For New Users, and try the examples as you read the manual. (The best way is corner an experienced and patient user and make him or her show you how things work!) The examples in the tutorial cover the basic commands you need to know in order to proceed further. The next step is to browse through the lists (one alphabetical and the other functional) of SAC commands in the Command Reference Manual, find a few that sound interesting, and try them. SAC2000 is fairly careful in checking for errors so you can't do much harm in experimenting. After you have become somewhat comfortable using SAC2000, you might want to return to this manual to get a better overview of the program. Be sure and read the section on SAC macros. They will definitely save you time when you get ready to do fairly complicated or repetitive analysis on large data sets. Eventually you will probably want to sit down and wade through the complete Command Reference Manual to get a complete picture of what you can do with SAC2000. It's better to defer that step until you have used the code for awhile.

2.3 Analysis Capabilities

SAC2000 User's Manual

Analysis Capabilities

Overview SAC2000 is logically divided into functional modules. Each functional module performs a related set of tasks. This section briefly describes the commands in each of these modules. The full command names are used in these descriptions. Most of the commands have convenient abbreviations. See the Command Reference Manual for details.

Executive Function Module Once you have successfully started SAC2000, you need to know how to get rid of it! This is done with the QUIT com-

mand. END, EXIT, and DONE are also allowed so you shouldn't have any problems. FUNCGEN lets you generate various functions in memory. It is useful for testing the other commands on known functions. DATAGEN lets you read sample data from three events (one local, one regional, and one teleseismic) into memory. This lets you play with some real seismic data while you are getting your own data converted to the SAC data file format. There are several commands that give you information about SAC2000: NEWS prints out general information about the latest version of SAC2000, HELP and SYNTAX give you information about a specific command, and REPORT gives you the current values of important parameters. SAC2000 has an extensive macro capability that is described in a later section. It lets you execute a set of SAC commands by putting them in a file. You can define arguments complete with default values, perform simple arithmetic calculations, store and retrieve information, and control the flow of command execution with if-tests and do-loops. MACRO executes a macro file. SETMACRO defines the search path to be used to find a macro file. INSTALLMACRO lets you make a macro available for use by anyone else on your system. You can store (SETBB), remove (UNSETBB), and retrieve (GETBB) information and do arithmetic calculations (EVALUATE) using the "blackboard." You can also save (WRITEBBF) and restore (READBBF) information in the blackboard into a disk file. Other commands that are useful in a macro include the ability to send a message to the terminal (MESSAGE), echo commands to the terminal (ECHO), and temporarily suspend the execution of a macro (PAUSE). You can write your own SAC command in FORTRAN or C routines that can be loaded into SAC2000 (LOAD), and executed thereafter just like an indigenous SAC command (see Notes and external_interface in aux/external). TRACE can be used to have SAC2000 trace header and blackboard variables, reporting to the screen when a variable changes values. TRANSCRIPT controls SAC2000's transcription capabilities, saving commands, and/or error messages, and/or warnings, and/or other output to a text file. COMCOR provides command correction. When SAC2000 detects an error during the course of executing a command, if this option is set, SAC2000 will allow the user to correct the

command and continue execution. CD changes SAC2000's current working directory. Finally you can execute operating system commands while running SAC2000 (SYSTEMCOMMAND) and reinitialize SAC2000 to its default state (INICM).

Condition Execution Module This module provide commands which control the flow of commands. These commands can only be called from within a macro, and are discussed in greater detail in the section on SAC macros. The commands in this module are IF, ELSEIF, ELSE, ENDIF, DO, WHILE, ENDDO, and BREAK.

Data File Module This module is used to read, write, and access SAC data files. These data files are described in detail in a later section. READ reads data files from disk into memory and WRITE writes the data currently in memory to disk. CUT defines how much of a data file is to be read. READERR controls errors that occur while files are being read and CUTERR controls errors due to bad cut parameters. Each data file has a header which describes the contents of the file. You can read and write these headers without the data using READHDR and WRITEHDR. You can also list the contents (LISTHDR), change values (CHNHDR), and copy header values from one file to the others in memory (COPYHDR). The SYNCHRONIZE command changes the headers in memory so that they all have the same reference time. You must first use this command before using the CUT command on files with different reference times. You can use READALPHA to read almost any alphanumeric data file directly into SAC2000. The read commands let you use wildcard characters to easily read in groups of files that contain the same pattern of characters. The WILD command controls certain aspects of this wildcard filename expansion. The SAC data file is stored in binary format for fast reading and writing. There is also an alphanumeric equivalent of this binary format. This is useful when transferring SAC data files from one kind of computer to another kind. CONVERT can be used to convert between the binary and alphanumeric formats. READCSS reads CSS 3.0 formatted flat files. Preferences for the way picks are read in are set in a preferences file but can be modified using the PICKAUTHOR and PICKPHASE command. WRITECSS writes the

data to flat files. WRITECSS is currently not working as comprehensively as READCSS. DELETECHANNEL allows you to delete one or more files from memory. READSDD and WRITESDD allow reading and writing of SDD data files.

Spectral Analysis Module You can do a discrete Fourier transform (FFT) and an inverse transform (IFFT). You can also compute the amplitude and unwrapped phase of a signal (UNWRAP). This is an implementation of the algorithm due to Tribolet. There is a set of Infinite Impulse Response filters (BANDPASS, BANDREJ, LOWPASS, and HIGHPASS), a Finite Impulse Response filter (FIR), an adaptive Wiener filter (WIENER), and two specialized filters (BENIOFF and KHRONHITE) used at LLNL. CORRELATE computes the auto- and cross-correlation functions. CONVOLVE computes the auto- and cross-convolution functions. The FFT and UNWRAP commands produce spectral data in memory. You can plot this spectral data (PLOTSP), write it to disk as “normal” data (WRITESP), and read it back in again (READSP). You can also perform integration (DIVOMEGA) and differentiation (MULOMEGA) directly in the frequency domain. HANNING applies a “hanning” window to each data file. HILBERT applies a Hilbert transform. ENVELOPE computes the envelope function using a Hilbert transform. KEEPAM keeps amplitude component of spectral files (of either the AMPH or RLIM format) in SAC memory.

Unary Operations Module The commands in this module perform some arithmetic operation on each data point of the signals in memory. You can add a constant (ADD), subtract a constant (SUB), multiply by a constant (MUL), or divide by a constant (DIV). You can square each data point (SQR), take the square root (SQRT), or take the absolute value (ABS). You can take the natural (LOG) or base 10 (LOG10) logarithm of each data point. You can also compute the exponential (EXP) or base 10 exponential (EXP10) of each data point. Lastly you can perform integration (INT) and differentiation (DIF).

Binary Operations Module These commands perform operations on pairs of data files. MERGE merges (concatenates) a set of files to the data in memory. ADDF adds a set of data files to the data in memory. SUBF

subtracts a set of data files from the ones in memory. MULF multiplies a set of data files by the data in memory. DIVF divides the data in memory by a set of files. BINOPERR controls errors that can occur during these binary operations.

Signal Correction Module These commands let you perform certain signal correction operations. RQ removes the seismic Q factor from spectral data. RTREND and RMEAN remove the linear trend and the mean from data respectively. RGLITCHES removes glitches and timing marks. TAPER applies a symmetric taper to each end of the data and SMOOTH applies an arithmetic smoothing algorithm. STRETCH upsamples data, including an optional interpolating FIR filter, while DECIMATE downsamples data, including an optional anti-aliasing FIR filter. You can interpolate evenly or unevenly spaced data to a new sampling interval using the INTERPOLATE command. LINEFIT computes the best straight line fit to the data in memory and writes the results to header blackboard variables. QUANTIZE converts continuous data into its quantized equivalent. REVERSE reverses the order of data points. Finally, you can rotate pairs of data components through a specified angle with the ROTATE command.

Event Analysis Module This module is used to pick seismic phases. An automatic phase picking algorithm can be applied using APK. You can also use PPK to pick phases using the graphics cursor. (PPK is described in the section on Graphics Capabilities). These picks can be saved in HYPO format using the OHPF (open HYPO pick file) and CHPF (close HYPO pick file) commands; WHPF writes auxiliary cards into the HYPO pick file. These picks can also be saved in a more general Alphanumeric format using the OAPF (open alphanumeric pick file) and CAPF (close alphanumeric pick file) commands. The picks are also saved in the headers.

Signal Measurement Module These commands measure and “mark” selected attributes about the data in memory. These marks are stored in the headers. MARKTIMES marks the data in memory with travel times from a velocity set. MARKPTP measures and marks the maximum peak to peak amplitude. MARKVALUE searches for and marks selected values in a signal. MTW sets the “measurement time window” option. When this

option is on, the measurements are made within this window only. Otherwise the measurements are made on the entire signal. MTW applies to the MARKPTP and MARKVALUE commands only. RMS computes the root mean square of the data within the measurement time window.

Instrument Correction Module This module currently contains only one command, TRANSFER. TRANSFER performs a deconvolution to remove one instrument response followed by a convolution to apply another instrument response. Over 40 predefined instrument responses are available. A general instrument response can also be specified in terms of its poles and zeros.

Neural Network Module This module has only one command, WRITENN, which writes data files to disk in neural net format.

XYZ Data Processing Module The commands in this module produce output that is a function of two input domains. SPECTROGRAM calculates a spectrogram using all of the data in memory. GRAYSCALE produces grayscale images of data in memory. CONTOUR produces contour plots of data in memory. ZLEVELS controls the contour line spacing in subsequent contour plots. ZLINES controls the contour linestyles in subsequent contour plots. ZTICKS controls the labeling of contour lines with directional tick marks. ZLABELS controls the labeling of contour lines with contour level values. ZCOLORS controls the color display of contour lines. IMAGE produces color sampled image plots of data in memory. SONOGRAM calculates a spectrogram equal to the difference between two smoothed versions of the same spectrogram.

Frequency-waveform Spectral Analysis Module Most of the command in this module are algorithms to extract wave field parameters from a suite of seismograms. ARRAYMAP produces a map of the array or “coarray” using all files in SAC memory. BBFK computes the broadband frequency-wavenumber (FK) spectral estimate, using all files in SAC memory. BEAMFORM computes the beam using all data files in SAC memory. GMTMAP generates a GMT (Generic Mapping Tools) map with station/event symbols using all the files in SAC memory and an event file specified on the command line.

Matlab Module This module provides an interface between SAC2000 and MATLAB, allowing users who have MATLAB the ability to utilize its facilities and m-files on SAC files. 3C launch a Matlab GUI for manipulating 3-component data. MAT allows processing of SAC data from within SAC2000 using the MATLAB engine. SETMAT allows user to specify directory to be added to MATLAB search path. CLOSEMAT closes MATLAB, quits the engine. When called from within MATLAB, it returns control to SAC2000.

Subprocesses A subprocess is like a small program within the larger SAC2000 program. It works like SAC2000 in many ways but the differences are such that it could not be included in the main program. Once invoked, only the commands within that subprocess plus a selected group of commands from the main SAC2000 program can be executed. The prompt changes to include the name of the subprocess. When done you can return to the main SAC2000 program using the QUITSUB command or terminate SAC2000 using the QUIT command.

Spectral Estimation Subprocess This subprocess is for the study of stationary random processes (i.e. noise.) Three spectral estimation techniques are available: the maximum entropy method, the maximum likelihood method, and the power density spectra method.

Signal Stacking Subprocess This subprocess is for performing signal stacking with delays. The delays can be static or dynamic. Two velocity models are available. The signals can be individually weighted. Traveltimes can be computed, or read from a file. A record section plot is also part of this subprocess.

2.4 Graphics Capabilities

SAC2000 User's Manual

Graphics Capabilities

Overview This section describes the graphics devices that are currently supported and then briefly describes the commands in each of the graphics functional modules.

Graphics Devices There are three graphics “devices” currently supported. The first one, SGF, is a general purpose device driver representing a large class of actual physical devices. The second, XWINDOWS, is a general windowing system running on most high-resolution, bit-mapped graphics workstations. The third, SUNWINDOW, is a windowing system that was available on the Sun in SunOS 4.X. Each device is described in more detail below. The number, type, and names of the graphics devices available on your system may be different from this list. Check with your system administrator or use the REPORT DEVICES command to determine which devices are available.

SGF stands for SAC Graphics File. A SAC Graphics File contains all the information needed to generate a single plot on any graphics device. (Using the current computer jargon, these are called graphics “metafiles.”) Each plot is stored in a separate file. The file names are of the form “Fnnn.SGF” where “nnn” is the plot number, beginning with “001”. You can control some features of this file name using the SGF command. Programs are available which can display these files to the terminal, merge several files into a single file, or produce an alphanumeric dump of a file for debugging. Programs are also available to convert these files to specific graphics devices such as the Apple Laserwriter, Houston Instruments pen plotter, etc. Some programs which handle SGF files are distributed with SAC2000 in either the bin or the utils directories. It is fairly easy to create such a conversion program. These programs and the SGF file format are described in the SGF Users Manual.

XWINDOWS (or X for short) is a windowing scheme developed under the industry-financed Athena project at MIT. X employs what is called a network model, where a single process or server controls the screen display. Other programs send requests to this server when they want to modify part of the screen. X is widely used on the graphics workstation and offers one of the best frameworks for developing portable window-based applications.

SUNWINDOW is a windowing system available on SunOS 4.X.

Each program contains its own set of procedures to control the screen. The two windowing systems are incompatible. You must be running under

one or the other at any given time. Benchmarks using the SAC graphics library do not show significant difference in graphics display speeds between these two windowing systems.

Graphics Control Module These commands control device selection and certain aspects of the display. `BEGINDEVICES` selects one or two graphics devices for plotting and `ENDDEVICES` deselects plotting to those devices. `ERASE` erases the graphics display area, `VSPACE` controls the maximum size and shape of plots, and `SGF` controls certain options for the SAC Graphics File device.

Graphics Action Module The commands in this module are mostly action-producing ones that create plots in various formats. `PLOT` plots each signal in memory on a separate plot. `PLOT1` plots a set of signals on a single plot with a common x axis and separate y axes. `PLOT2` plots a set of signals on a single plot with common x and y axes (i.e. an overlay plot). `PLOTPK` produces a plot for the picking of arrival times, seismic phases, coda, etc. The format is similar to that of `PLOT1`. A cursor is used to do the picking. The picks go into the header and can also be written into a `HYPO` pick file (`OHPF`) or an alphanumeric pick file (`OAPF`). `PLOTPM` generates a “particle-motion” plot on pairs of signals. `FILEID` controls the display of a file identification and `FILENUMBER` controls the display of file numbers on the sides of plots. `PICKS` controls the display of time picks on these plots. `SETDEVICE` lets you select a default graphics device to be used when plotting. `PLOTG` annotates SAC2000 plots and creates figures using cursor. `PLOTALPHA` reads alphanumeric data files on disk into memory and plots the data to the current output device. `PLOTDY` creates a plot with error bars. `PLOTXY` plots one or more data files versus another data file.

Graphics Environment Module The commands in this module are mostly parameter-setting ones that control various parts of the plots produced by the Graphics Action Module. `XLIM` and `YLIM` control the plot limits for the x and y axes. `XVPORT` and `YVPORT` control the location of the plot within the plotting area. You can specify a title (`TITLE`), x and y axes labels (`XLABEL` and `YLABEL`, and a set of general plot labels (`PLA-`

BEL). There are several commands that control the displaying of the data itself: LINE controls linestyle selection, SYMBOL controls symbol plotting, and COLOR controls color selection. GTEXT controls the quality and font of text used in plots and TSIZE controls the text size attributes. If you are using a multi-windowing workstation, you can use the WINDOW command to set the location and shape of the graphics windows and the BEGINWINDOW command to select a specific graphics window for plotting. BEGINFRAME turns off automatic new frame actions between plots and ENDFRAME resumes automatic new frame actions. Combined with other graphics commands (especially XVPORT and YVPORT), these two commands can be used to create fairly complicated plots. XLIN and XLOG turn on linear and logarithmic scaling for the x axis. YLIN and YLOG do the same for the y axis. You can also use the commands LINLIN, LINLOG, LOGLIN, and LOGLOG to set the scaling for both axes with one command. XDIV and YDIV control the spacing between labeled divisions while XFUDGE and YFUDGE change the “fudge factors” on the two axes. AXES and TICKS control the location of labeled axes and tick marks. GRID and BORDER control the plotting of grid lines and a surrounding border. There are also commands (XGRID and YGRID) that let you independently control gridding on either axis. There are several commands which control the display of logarithmic axes: XFULL and YFULL control the plotting of full logarithmic decades, LOGLAB controls the plotting of secondary labels, and FLOOR puts a minimum value on logarithmically scaled data. LOADCTABLE allows the user to select a new color table for use in image plots. WAIT tells SAC2000 whether or not to pause between plots. WIDTH controls line-width selection for graphics devices. NULL controls the plotting of null values. Finally, the QDP command controls the “quick and dirty plot” option. I’ll let you look that one up in the Command Reference Manual!

2.5 SAC Macors

SAC2000 User’s Manual

SAC Macros

Overview

A SAC macro is a file that contains a set of SAC commands to be executed together. As well as regular commands and inline functions, a SAC macro file can contain references to SAC header variables and blackboard variables that are evaluated and substituted into the command before it is executed. SAC macros can also have arguments that are evaluated as the macro is executed. Control flow features such as “if tests” and “do loops” are also available. These features let you control and alter the order of execution of commands within a macro. All of these features are discussed later in this section.

A Simple Example Assume that you have a set of commands that you execute repeatedly. A macro file is the obvious solution. Simply fire up your favorite text editor, put the commands into a file, and then have SAC2000 execute them using the MACRO command. Lets say you wanted to repeatedly read the same three files, multiply each file by a different value, take the fourier transform, and plot the amplitude responses to a set of SAC Graphics Files. The macro file would look like this:

```
* This certainly is a simple little macro. READ ABC DEF XYZ MUL
4 8 9 FFT BG SGF PSP AM
```

Assume the file is called MYSTUFF and that it and the data are in the directory to which you are currently attached. To execute the macro from SAC2000 type:

```
u: MACRO MYSTUFF
```

Note that commands in a macro file are not normally echoed to the terminal as they are executed. You can use the ECHO command to turn command echoing on if you wish. Also note that an asterisk in the first column of a line denotes a comment line and is not processed by SAC2000.

Order Dependent Arguments The above example while simple is also very inflexible. If you wanted to read a different set of files or use a different set of multiplicative values you have to edit the file. Allowing macros to have arguments that you enter at execution time greatly increases their flexibility. We will modify the previous macro to accept the names of the files as arguments:

```
READ $1 $2 $3 MUL 4 8 9 FFT BG SGF PSP AM
```

The dollar sign (“\$”) is used to delineate arguments in a macro file. \$1 is the first argument, \$2 the second, \$3 the third, and so on. To execute this modified macro from SAC2000 type:

```
u: MACRO MYSTUFF ABC DEF XYZ
```

The token “ABC” is substituted wherever the “\$1” token is found. Also “DEF” and “XYZ” are substituted for “\$2” and “\$3” respectively. To execute the same macro with a different set of files only the execute line changes:

```
u: MACRO MYSTUFF AAA BBB CCC
```

Keyword Driven Arguments Keyword driven arguments let you enter arguments in any order and also makes the body of a macro easier to understand. This becomes increasingly important as the number of arguments and the size of the macro increase. Lets again modify our example to accept a list of files and also a list of multiplicative values:

```
$KEYS FILES VALUES READ $FILES MUL $VALUES FFT BG SGF  
PSP AM
```

This simple change has increased both the flexibility and the readability of the macro. The first line says that there are two keywords, one called “FILES” and the other called “VALUES”. To execute it you could type:

```
u: MACRO MYSTUFF FILES ABC DEF XYZ VALUES 4 8 9
```

Since the order of the arguments is no longer important you could also type:

```
u: MACRO MYSTUFF VALUES 4 8 9 FILES ABC DEF XYZ
```

This macro doesn’t limit you to reading in only three files. It would work equally well for two or 10 files as long as the number of values match the number of files.

Default Argument Values There are times when you have a macro where some arguments often (but not always) have the same value from one execution to the next. Providing default values for such arguments eliminates the need to enter the same values each time but allows you the flexibility to enter them when needed. This is demonstrated in the next example:

```
$KEYS FILES VALUES $DEFAULT VALUES 4 8 9 READ $FILES  
MUL $VALUES FFT BG SGF PSP AM
```

The second line in the macro specifies a default value to be used for the variable “VALUES” if you don’t enter one on the execute line:

```
u: MACRO MYSTUFF FILES ABC DEF XYZ
```

If you wanted to use a different set of values you could type:

```
u: MACRO MYSTUFF VALUES 10 12 3 FILES ABC DEF XYZ
```

Argument Querying If you fail to enter a value for an argument on the execute line and it has no default value, SAC2000 will ask you to enter a value from the terminal. Using the macro in the previous section, assume that you forgot to enter the filelist:

```
u: MACRO MYSTUFF s: FILES? u: ABC DEF XYZ
```

Note that SAC2000 did not query for “VALUES” because it had a default value. The timing of this query can sometimes be used to your advantage. SAC2000 does not query for a value until it first tries to evaluate the argument and finds that it has no default or input value. This allows part of the macro to execute showing you some partial results before asking you to enter values for an argument.

Blackboard Variables; SAC2000 has a blackboard feature that can be used to temporarily store and retrieve information. A blackboard entry consists of a name and a value. Blackboard entries are created using the SETBB and EVALUATE commands. The value of a blackboard variable can be obtained using the GETBB command. You can also substitute the value of a blackboard variable directly in other commands by preceeding its name with a percent sign (“%”) as shown below:

```
u: SETBB C1 2.45 u: SETBB C2 4.94 u: BANDPASS CORNERS %C1
%C2
```

Now lets see how blackboard variables can be used in macros. (You are probably getting tired of endless variations on our original macro, but we are almost done with it.) Assume that only the first value was a variable, i.e. the other values could be calculated from the first as shown below:

```
$KEYS FILES VALUE1 $DEFAULT VALUE1 4 READ $FILES EVAL-
UATE TO VALUE2 $VALUE1 * 2 EVALUATE TO VALUE3 %VALUE2
+ 1 MUL $VALUE1 %VALUE2 %VALUE3 FFT BG SGF PSP AM
```

Now only the first value is input to the macro and only if it differs from

the default value:

```
u: MACRO MYSTUFF VALUE1 6 FILES ABC DEF XYZ
```

Header Variables SAC header variables can also be evaluated and substituted directly in commands much like blackboard variables. You must specify which file (by name or number) and which variable to be evaluated. You must precede this specification with an ampersand (“&”) and you must separate the file and variable with a comma as shown below:

```
u: READ ABC u: EVALUATE TO TEMP1 &ABC,A + 10 u: EVAL-  
UATE TO TEMP2 &1,DEPMAX * 2 u: CHNHDR T5 %TEMP1 u: CHN-  
HDR USER0 %TEMP2
```

In the above example a file is read in and several temporary blackboard variables are calculated using header variables from the file itself. The first header reference is by file name and the second by file number. New header variables are then defined using these blackboard variables.

Concatenation You can append or prepend any text string to a macro argument, blackboard variable, or header variable. To prepend simply concatenate the text string with the argument or variable. To append you must repeat the delimiter (\$, %, or &) after the argument or variable and before the text string. Sounds confusing? See the examples below for some clarification:

Assume that the macro argument STATION has the value “ABC”. Then value of “\$STATION\$.Z” would be “ABC.Z”.

Assume that the blackboard variable TEMP has the value “ABC”. Then value of “XYZ%TEMP” would be “XYZABC” and the value of “%TEMP%XYZ” would be “ABCXYZ”.

Assume that the header variable KA for file Z has the value “IPU0”. Then value of “(&Z,KA&)” would be “(IPU0)”.

Nesting and Recursion When a macro can call another macro which can call another macro, etc., this is often referred to as nesting. When one macro calls another, the second macro is said to be operating at a new (lower) level of execution. The top level of execution is always interactive input from the terminal. When a macro can call itself, then it is said to be recursive. The SAC macro capability supports nesting but not recursion. SAC2000 does

not check to ensure that macro calls are not recursive. It is the responsibility of the user to make sure a macro is not directly or indirectly calling itself.

Interrupting a MACRO There are occasions when you need to temporarily interrupt the execution of a macro, enter a few commands from the terminal, and then continue executing the macro. This can be done in SAC2000 using the pause and resume feature. When SAC2000 sees a \$TERMINAL in a macro it temporarily stops reading commands from the macro, changes its prompt to include the name of the macro, and starts prompting for commands from the terminal. Then when SAC2000 sees a \$RESUME entered from the terminal it stops reading commands from the terminal and begins reading from the macro starting at the next line (the one after the \$TERMINAL.) If you don't want to continue executing the commands in the macro you can type a \$KILL from the terminal. SAC2000 will then close the macro file and return to the previous level of execution, normally interactive input from the terminal. You can have more than one \$TERMINAL in a macro.

If Tests This feature lets you alter the order of commands being executed from a macro file. The syntax is similiar but not identical to the if-then-else clause in F77:

```
IF expr commands ELSEIF expr commands ELSE commands ENDIF
```

In the above clause expr is a logical expression of the form:

token op token

where token is a constant, macro argument, blackboard variable, or a header variable and op is one of the following logical operators:

GT—GE—LE—LT—EQ—NE

The tokens are converted to floating point numbers before the logical expression is evaluated. The maximum number of nested if clauses is currently set at 10. The ELSEIF and ELSE elements are optional. There is no limit of the number of ELSEIF elements in an if clause. Note that there are no parentheses around a logical expression and no THEN keyword ending the IF and ELSEIF elements as in F77. An example is given below:

```
READ $1 MARKPTP IF &1,USER0 GE 2.45 FFT PLOTSP AM ELSE  
MESSAGE "Peak to peak for \"$1 below threshold." ENDIF
```

In this example a file is read into memory and the maximum peak to peak amplitude is measured. (MARKPTP stores this amplitude into the header variable USER0.) If this amplitude is above a certain value, a fourier transform is calculated and the amplitude response is plotted. If not a message is sent to the terminal.

Do Loops These features let you easily repeat a set of commands. You can execute a set of commands a fixed number of times, for each element in a list, or until a condition has been met. You can also break out (prematurely terminate the execution) of a do loop. The syntax for this group is summarized below:

```
DO variable = start, stop, {,increment} commands ENDDO
DO variable FROM start TO stop { BY increment} commands ENDDO
DO variable} LIST} entrylist} commands ENDDO
DO variable WILD {DIR name} entrylist commands ENDDO
WHILE expr commands ENDDO
BREAK
```

where

variable is the name of the do loop variable. Its current value while the do loop is executing is stored as a macro argument and may be used in the body of the do loop (i.e., the commands) by preceeding its name with a dollar sign.

start is the starting value for the do loop variable. It must be an integer.

stop is the stopping value for the do loop variable and must also be an integer.

increment is the optional increment in the do loop variable. If omitted, the default value is set to 1.

entrylist is a space delimited list of values that the do loop variable is to have. These may be integers, floating point numbers, or character strings. In the DO WILD case, the entrylist consists of character strings containing both regular and wildcard characters. This entrylist is expanded into a list of files that match the character strings before the do loop is executed.

expr is a logical expression as described in the section on if tests.

The maximum number of nested do loops is currently set at 10. Examples of each of these do loops are given below.

Do Loop Examples

The first macro applies the DIF command to a data file (to prewhiten the data), performs a fourier transform on the data, and then applies the DIVOMEGA command to remove the effect of the prewhitening. Sometimes it is necessary to differentiate the data more than once before doing the transform. This can be handled with a do loop:

```
$KEYS FILE NPREW $DEFAULT NPREW 1 READ \ $FILE DO J =
1 , $NPREW DIF ENDDO FFT AMPH DO J = 1 , $NPREW DIVOMEGA
ENDDO
```

Notice the use of a default value for the order of the prewhitening. In the second example, particle motion plots are produced for five different two second time windows on the same data file:

```
READ ABC SETBB TIME1 0 DO TIME2 FROM 2 TO 10 BY 2 XLIM
%TIME1 $TIME2 TITLE 'Particle Motion from %TIME1 to $TIME2$'
PLOTSM SETBB TIME1 $TIME2 ENDDO
```

(Why is a dollar sign needed after TIME2 in the TITLE command?) In the next example, a macro called PREVIEW exists that performs a set of commands on a single data file. A new macro is created with several nested do loops to run PREVIEW on a predefined group of data files:

```
DO STATION LIST ABC DEF XYZ DO COMPONENT LIST Z N E
MACRO PREVIEW $STATION$. $COMPONENT$ ENDDO ENDDO
```

In the next example we modify the previous one to now process all of the files in the directory called “MYDIR” that end in the letters “.Z”:

```
DO FILE WILD DIR MYDIR *.Z MACRO PREVIEW $FILE ENDDO
```

The last (somewhat artificial) example has three arguments. The first is the name of a data file, the second a multiplicative constant, and the third a threshold value. The macro reads the data file into memory, and multiplies each data point by the constant until the maximum value is below the threshold:

```
READ $1 WHILE &1,DEPMAX GT $3 MUL $2 ENDDO
```

Another version of this macro illustrates the BREAK statement:

```
READ $1 WHILE 1 GT 0 DIV $2 IF &1,DEPMAX GT $3 BREAK
ENDIF ENDDO
```

This WHILE loop in this macro is an example of a “do forever” loop which can only be terminated by a BREAK statement. (This version of the macro has a flaw. What happens if the maximum value is already below the threshold?) The BREAK statement terminates the execution of the do loop where the statement appears.

Executing Other Programs From SAC Macros

You can execute other programs from inside a SAC macro. You can pass an optional execution line message to the program. If the program is interactive, you can also send input lines to it. The syntax for this feature is given below:

```
RUN program {message} {inputlines} ENDRUN
```

Macro arguments, blackboard variables, header variables, and inline functions may be used in the above lines. They are all evaluated before the program is executed. When the program completes the SAC macro resumes at the line following the {\am ENDRUN} line.

Macro Search Path

When you request a macro, SAC2000 searches for it as follows:

- in the current directory.

- in the directories specified in the SETMACRO command.

- in the global macro directory that is maintained by SAC2000.

The global macro directory contains macros meant to be used by everyone on your system. Use the INSTALLMACRO command to install a macro in this directory. You may also specify the absolute or relative pathname of a macro that is not in this search path.

Execution Line Macro

If you enter the name of a macro on the SAC execution line, SAC2000 will automatically execute the commands in this macro file before prompting you for input. If the macro has arguments they may appear after the macro name. You must specify the full pathname of this macro if it is not in the current directory.

The Escape Character

There may be times when you need to use a dollar sign or a percent sign in a command and not have SAC2000 interpret it as a macro argument or blackboard variable entry. To do this you precede the special character with another special character, called the escape character. The escape character is an at sign (“@”). The special characters that must be treated in this way are:

\$: The macro argument expansion character.

?: The blackboard variable expansion character.

&: The header variable expansion character.

@: The escape character itself.

(: The inline function starting character.

): The inline function terminating character.

More about the inline function delimiters in the next section.

Acknowledgements

The concept of blackboard variables are due to Dave Harris. The “if test” and “do loop” features were developed by Mandy Goldner.

2.6 Inline Functions

SAC2000 User’s Manual

Inline Functions

Overview

An inline function is one that is enclosed in parenthesis and placed within a regular SAC command. The inline function is evaluated and its resulting value replaces the function in the SAC command before the command is executed. There are three general classes of inline functions:

embedded arithmetic functions which begin with a number and have the name of the function embedded in the argument list.

regular arithmetic functions which begin with the function name and are followed by zero or more arguments.

character string manipulation functions which begin with the function name and are followed by zero or more arguments.

Inline functions can be placed inside other inline functions. This is referred to as nesting. There is a current limit to this nesting of 10 levels. Macro arguments, blackboard variables, and header variables can be used as arguments to inline functions. They are inserted in inline functions using the same syntax as in regular SAC commands.

Embedded Arithmetic Functions An embedded arithmetic function is much like (but alas not identical to) the right hand side of a FORTRAN arithmetic statement. It is of the general form:

(number operator number ...) where number is a numeric value and operator is one of the following arithmetic operators:

+, -, *, /, ** Lets look at a simple example:

u: SETBB A (4 + 7 / 3) s: ==> SETBB A 3.666667

Notice that whenever an inline function is found in a command, SAC2000 prints the processed (i.e. evaluated) command to the terminal. This allows you to see the command that was actually executed. This example also illustrates two of the differences between inline functions and FORTRAN statements:

All numbers are treated as real and all arithmetic is done in floating point.

There is no implied precedence among operators. Calculations are done in order from left to right.

In the above example the real numbers 4.0 and 7.0 are first added together and then divided by the real number 3.0 to get the result. Embedded functions can be nested to achieve a different order of computation:

u: SETBB A (4 + (7 / 3)) s: ==> SETBB A 6.333333

In this example the division would be performed first. Also notice the space between the plus sign and the second left parenthesis. This is necessary in order for SAC2000 to parse the command properly. In general it is wise to place spaces around all arguments, operators, and nested parentheses.

Regular Arithmetic Functions There are currently 20 regular arithmetic functions available. They correspond to the arithmetic functions found in the EVALUATE command. Each of these functions is described below. Some examples are given at the end of this subsection.

ADD SYNTAX: (ADD v1 v2 ... vn)
 PURPOSE: Add (sum) a set of numbers.
 SUBTRACT SYNTAX: (SUBTRACT v1 v2 ... vn)
 PURPOSE: Subtract a set of numbers.
 MULTIPLY SYNTAX: (MULTIPLY v1 v2 ... vn)
 PURPOSE: Multiply a set of numbers.
 DIVIDE SYNTAX: (DIVIDE v1 v2 ... vn)
 PURPOSE: Divide a set of numbers.
 SQRT SYNTAX: (SQRT v)
 PURPOSE: Take the square root of a number.
 EXP SYNTAX: (EXP v)
 PURPOSE: Exponentiate a number.
 ALOG SYNTAX: (ALOG v)
 PURPOSE: Take the natural logarithm of a number.
 POWER SYNTAX: (POWER v)
 PURPOSE: Raise a number to its power of 10.
 ALOG10 SYNTAX: (ALOG10 v)
 PURPOSE: Take the log to base 10 of a number.
 SINE SYNTAX: (SINE v)
 PURPOSE: Take the sine of a number.
 ARCSINE SYNTAX: (ARCSINE v)
 PURPOSE: Take the arcsine of a number.
 COSINE SYNTAX: (COSINE v)
 PURPOSE: Take the cosine of a number.
 ARCCOSINE SYNTAX: (ARCCOSINE v)
 PURPOSE: Take the arccosine of a number.
 TANGENT SYNTAX: (TANGENT v)
 PURPOSE: Take the tangent of a number.
 ARCTANGENT SYNTAX: (ARCTANGENT v)
 PURPOSE: Take the arctangent of a number.
 INTEGER
 SYNTAX: (INTEGER v)
 PURPOSE: Convert a number to an integer.

PI

SYNTAX: (PI v)

PURPOSE: Return the value of pi.

MINIMUM

SYNTAX: (MINIMUM v1 v2 ... vn)

PURPOSE: Compute the minimum value of a set of numbers.

MAXIMUM

SYNTAX: (MAXIMUM v1 v2 ... vn)

PURPOSE: Compute the maximum value of a set of numbers.

ABSOLUTE

SYNTAX: (ABSOLUTE v)

PURPOSE: Take the absolute value of a number.

Lets look at several examples. To normalize a set of data files so that the maximum absolute value of any data point in the set is unity:

```
u: READ FILE1 FILE2 FILE3 FILE4 u: SETBB A (MAX &1,DEP-
MAX &2,DEPMAX &3,DEPMAX &4,DEPMAX) s: ==i SETBB A 1.87324
u: SETBB B (MIN &1,DEPMIN &2,DEPMIN &3,DEPMIN &4,DEPMIN)
s: ==i SETBB B -2.123371 u: DIV (MAX %A (ABS %B)) s: ==i DIV
2.123371
```

This could have been done in single command, without using intermediate blackboard variables, by nesting the inline functions properly, but this way is more readable. (It also fits on this page better!) In the next example, we need to calculate the tangent of an angle that has already been stored in the blackboard in degrees:

```
u: GETBB ANGLE s: ANGLE = 45.0 u: SETBB VALUE (TAN
(DIVIDE (MULTIPLY (PI) %ANGLE%) 180.)) s: ==i SETBB VALUE
1.00000
```

Having the name of the function as the first token in the function (called prefix notation) makes it easier for SAC2000 to parse the function. In some cases, especially the arithmetic ones, they are difficult to read. We can rework the above example into a more natural form by intermixing regular (prefix) and embedded arithmetic functions:

u: SETBB VALUE (TAN ((PI) * %ANGLE / 180.)) s: ==, SETBB
VALUE 1.00000

Why is the percent sign needed after ANGLE in the first example and is not needed in the second example?

Miscellaneous Arithmetic Functions

There is currently only one miscellaneous arithmetic function, GET-TIME, which returns the time offset (in seconds) relative to file begin time, for the first data point meeting the selection criteria.

GETTIME

SYNTAX: (GETTIME MAX—MIN [value])

PURPOSE: Returns the time (in seconds) for the first file in memory, of the datapoint having the requested value. If no value is specified, MAX returns the time of the file's first data-point having a value greater than or equal to DEPMAX; MIN returns the time of the file's first data-point having the value less than or equal to DEPMIN. Specifying a value controls the value of the data-point being searched for.

Lets look at some examples. To return the time in seconds of the first data-point greater than or equal to the file's maximum amplitude, for the file FILE1:

u: READ FILE1 FILE2 FILE3 FILE4 u: SETBB MAXTIME (GET-
TIME MAX) s: ==, SETBB MAXTIME 41.87

The file's first data-point having a value greater than or equal to the file's maximum amplitude is located 41.87 seconds into the file.

To locate the time of the first data-point less than or equal to the value 123.45:

u: SETBB VALUETIME (GETTIME MIN 123.45) s: ==, SETBB
VALUETIME 37.9

The first data-point in the file having a value less than or equal to 123.45 occurs at 37.9 seconds into the file.

String Functions

There are currently seven string manipulation functions. Each of these functions is described below. Some examples are given at the end of this subsection.

CHANGE

SYNTAX: ({CHA}NGE} s1 s2 s3)

PURPOSE: Change one text string (s1) to another (s2) in a third text string (s3).

DELETE

SYNTAX: ({DEL}ETE s1 s2)

PURPOSE: Delete a text string (s1) within another text string (s2).

BEFORE

SYNTAX: ({BEF}ORE s1 s2)

PURPOSE: Return the portion of a text string (s2) that occurs before another text string (s1).

AFTER SYNTAX: ({AFT}ER s1 s2)

PURPOSE: Return the portion of a text string (s2) that occurs after another text string (s1).

SUBSTRING

SYNTAX: ({SUBS}TRING n1 n2 s)

PURPOSE: Return substring with characters n1 through n2 of text string (s).

CONCATENATE

SYNTAX: ({CONC}ATENATE s1 s2 ... sn})

PURPOSE: Concatenate (i.e., place end to end) one or more text strings.

REPLY

SYNTAX: ({REP}LY s1) ({REP}LY s1[default_value])

PURPOSE: Send a message to the terminal and get a reply. If a default value is specified, the user can simply hit return at the prompt and SAC2000 will take the default value.

The following examples show the use of several of the above functions. To use the station and event names in the title of a plot:

```
{u:} FUNCGEN SEISMOGRAM {u:} TITLE '(CONCATENATE 'Seis-
mogram of ' &1,KEVNM ' ' &1,KSTNM )' {s:} ==i TITLE 'Seismogram
of K8108838 CDV'
```

The previous example shows several features (and potential difficulties) of the inline string functions. This example of the CONCATENATE func-

tion has four arguments. The first argument has spaces in it so has to be enclosed in either (single or double) quotes. The second and fourth arguments have no spaces in them so they don't need quotes. The third argument consists of a single space so that the event and station names don't run together. Finally the quotes around the inline function itself are required because of the syntax of the title command.

The next example uses the SUBSTRING function to extract the month of the event and store it into a blackboard variable.

```
{u:} FUNCGEN SEISMOGRAM {u:} SETBB MONTH (SUBSTRING
1 3 '&1,KZDATE&') {s:} ==i SETBB MONTH MAR
```

Why are the quotes needed around the header variable KZDATE?

The next example uses the REPLY function to interactively control the processing of a set of data files:

```
{u:} DO FILE LIST ABC DEF XYZ {u:} _ READ \ $FILE {u:} _ DO
J FROM 1 TO 10 {u:} --- MACRO PROCESSFILE {u:} --- PLOT {u:}
--- SETBB RESPONSE (REPLY "Enter -1 to stop, 0 for next file, --- 1 for
same file: ") {u:} --- IF \%RESPONSE LE 0 THEN {u:} ----- BREAK {u:}
--- ENDIF {u:} _ ENDDO {u:} _ IF \%RESPONSE LT 0 THEN {u:} ---
BREAK {u:} _ ENDIF {u:} ENDDO
```

The outer do loop reads one file in at a time from a list. The inner loop calls a macro to process this file. The inner loop executes up to 10 times. After each execution of the processing macro, the file is plotted, a message is sent to the terminal, and the reply is saved in a blackboard variable. The first if tests this variable to see if the inner processing loop should be terminated (by executing the BREAK statement) or continued. The second if tests this same variable to see if the loop on each data file should be terminated or continued. If only one if test is needed, the REPLY function could be substituted directly into the if test and a blackboard variable would not be needed.

The next example shows REPLY with a default value:

```
{u:} SETBB BBDAY (REPLY "Enter the day of the week: [Monday]")
```

When this function is executed, the quoted string will appear on the screen, prompting the user for input. If the user types a string, SAC2000 will

put the string that the user entered into the blackboard variable BBDAY. If the user simply hits return, SAC2000 will put the default value (in this case, the string “Monday”) into BBDAY.

2.7 Blackboard Variables

SAC2000 User’s Manual

Accessing Blackboard Variables In Home Grown Software

Overview The blackboard is a feature that can be used to temporarily store and retrieve information while inside SAC2000. Blackboard variables can also be saved in a disk file using the WRITEBBF command and later restored into SAC2000 using the READBBF command. There are four functions in the sac.a library which allow the user to read and write blackboard variables in home grown software. There is also a function called UNSETBBV which deletes a variable from the blackboard. The sac.a library is available on the SAC ftp site. (Note: On the DEC Alpha, sac.a is broken in to two libraries: sac.a and sac2.a, both are necessary.)

Blackboard variables in SAC2000 A blackboard entry consists of a name and a value. Blackboard entries are created using the SETBB and EVALUATE commands. The value of a blackboard variable can be obtained using the GETBB command. You can also substitute the value of a blackboard variable directly in other commands by preceeding its name with a percent sign (“%”) as shown below:

```
u: SETBB C1 2.45 u: SETBB C2 4.94 u: BANDPASS CORNERS %C1
%C2
```

Now lets see how blackboard variables can be used in macros. In the following example, the first value is a variable, and the other values are calculated from the first:

```
$KEYS FILES VALUE1 $DEFAULT VALUE1 4 READ $FILES EVAL-
UATE TO VALUE2 $VALUE1 * 2 EVALUATE TO VALUE3 %VALUE2
+ 1 MUL $VALUE1 %VALUE2 %VALUE3 FFT BG SGF PSP AM
```

You can append or prepend any text string to a blackboard variable. To prepend simply concatenate the text string with the variable. To append

you must repeat the delimiter (%) after the variable and before the text string.

Examples: Assume that the blackboard variable TEMP has the value “ABC”. Then value of “XYZ%TEMP” would be “XYZABC” and the value of “%TEMP%XYZ” would be “ABCXYZ”.

More information on the use of blackboard variables in SAC macros is given in the section on SAC macros.

Blackboard I/O in SAC2000 There are four SAC commands which are used to read and write blackboard variables and to set and get blackboard variable values. These are READBBF, WRITEBBF, GETBBV, and SETBBV. These are SAC commands which can be called at the SAC prompt or within a SAC macro. For details, use the HELP command.

Blackboard I/O in your own C or FORTRAN programs The SAC library sac.a (on the DEC Alpha there are two: sac.a and sac2.a) which is available on the SAC2000 ftp site, contains five blackboard I/O routines which you can call within your home grown C or FORTRAN programs. These routines: read the blackboard variable files (READBBF), write blackboard variable files (WRITEBBF), get the current values of blackboard variables (GETBBV), set new values of blackboard variables (SETBBV), and delete a variable from the blackboard (UNSETBBV).

An Example The following is a short FORTRAN program that reads in a blackboard variable file gets the values of a few variables, sets the value of a new one, and then writes the file back to disk:

```
PROGRAM BBEXAMPLE
  CHARACTER KSTRING*(80), KTEMP*(16) CALL READBBF('bbfile',NERR)
  CALL GETBBV('STRING',KSTRING,NERR) CALL GETBBV('VALUE',KTEMP,NERR)
  READ(KTEMP,'(G16.5)')VALUE CALL DOSOMETHING(KTEMP,VALUE,RESULT)
  WRITE(KTEMP,'(G16.5)')RESULT CALL SETBBV('RESULT',KTEMP,NERR)
  CALL WRITEBBF('bbfile',NERR) END
```

The values of all blackboard variables are stored as character strings. Line 2 defines two character variables that are used in accessing these blackboard variables. Line 3 reads in a file called “bbfile”. Lines 4 and 5 access two variables from that file, one called “STRING” and one called “VALUE”.

The first is stored in the character variable “KSTRING” and the second in “KTEMP”. Line 6 converts “KTEMP” to a floating point number and stores it in “VALUE”. You must do this kind of conversion if you want to use a blackboard variable as a floating point or integer number in your program. Line 7 calls a subroutine which performs some calculation using these two variables and returns a floating point result. Line 8 converts this result into a character variable and line 9 stores it in the blackboard using the name “RESULT”. Line 10 writes the blackboard back to disk in the same file.

Special Note The names of blackboard variables are converted to uppercase before being stored or retrieved. This means that you can use either uppercase or lowercase in your program. However, the name of the blackboard variable file must be given exactly as it appears on disk. No case conversion is done on file names.

Also, when compiling/linking your code, it may be necessary to include ‘-lX11 -lm’ on the compile/link line in order to access the X windows and math libraries respectively.

Below is a C program which performs the same functions as the FORTRAN program above.

```
#include <stdio.h> #include <stdlib.h>
main () { float result , value ; long int nerr ; char kstring[ 81 ] , ktemp[
17 ] ;
    readbbf ( “bbfile” , &nerr , 6 ) ; getbbv ( “string” , kstring , &nerr , 6 ,
80 ) ; getbbv ( “value” , ktemp , &nerr , 5 , 16 ) ; value = atof ( ktemp ) ;
result = dosomething ( ktemp , value ) ; sprintf ( ktemp , “%16.5g” , result
) ; setbbv ( “result” , ktemp , &nerr , 6 , 16 ) ; writebbf ( “bbfile” , &nerr ,
6 ) ; }
```

Notice that in C, more parameters are required in the function calls than in FORTRAN. This is because unlike C, FORTRAN implicitly passes string length specifiers for each string in the parameter list. These specifiers are at the end of the parameter list, and are declared as INTEGER*4 or long int. Notice also that the values passed as string length specifiers do not include the null terminator ‘\0’.

2.8 Using SACIO

SAC2000 User's Manual

Reading And Writing SAC Data Files In Home Grown Software

Overview You can write your own stand-alone codes in C or FORTRAN and use SAC2000 library routines to handle I/O of SAC formatted data files. The library is called sacio.a, and is available on the SAC ftp site.

```
/******
```

Note: when compiling/linking your code, it may be necessary to include '-lX11 -lm' on the compile/link line in order to access the X windows and math libraries respectively.

```
/******
```

There are two routines in the SAC library that you can use to read SAC data files into your C or FORTRAN program: RSAC1 reads evenly spaced files RSAC2 reads unevenly spaced and spectral files.

There is a set of routines that let you get the values of header variables after the file has been read: GETFHV gets float or REAL header variables GETIHV gets character strings enumerated as int or INTEGER header variables GETKHV gets character string header variables GETLHV gets LOGICAL header variables (declared as long in C) GETNHV gets int or INTEGER header variables.

There is a like set of routines that let you set the values of header variables currently in memory: SETFHV sets float or REAL header variables SETIHV sets character strings enumerated as int or INTEGER header variables SETKHV sets character string header variables SETLHV sets LOGICAL header variables (declared as long in C) SETNHV sets int or INTEGER header variables.

There are three routines used to write SAC data files to disk: WSAC1 writes evenly spaced files WSAC2 writes unevenly spaced and spectral files WSAC0 writes more comprehensive header files than the other two (it requires the use of the next routine: NEWHDR).

Finally, there is a routine called NEWHDR which initializes a SAC header to undefined values. This is used in conjunction with WSAC0.

FORTRAN call statements and C prototypes of these routines are documented in the appendix.

A few examples are given below, including explanations.

RSAC1 FORTRAN —

```
PROGRAM RSAC PARAMETER (MAX=1000) DIMENSION YARRAY(MAX)
CHARACTER*10 KNAME KNAME='FILE1' CALL RSAC1(KNAME,YARRAY,NLEN,BEG,DEL,M
IF(NERR.GT.0)GO TO 8888 CALL DOIT(YARRAY,NLEN) 8888 CON-
TINUE END
```

Line 2 defines the maximum size of the data array. Lines 4 and 5 define the name of the data file to be read. This name must be a blank filled character variable. Line 6 calls RSAC1, which opens the file, reads the header, copies the data to the data array, and closes the file. RSAC1 also passes back the number of points read, the begin time, and the sampling interval. Line 8 calls your own subroutine to process this data in some fashion (otherwise why are we doing this at all!)

Error Checking Line 7 checks the error return flag to make sure the data was read properly. Most SAC subroutines contain this error return flag and all of them follow the same convention. The flag is zero if no error occurred, positive if a fatal error occurred that prevented successful completion, and negative if a warning condition occurred for which corrective action was taken. For example, if FILE1 does not exist, NERR is set to 108 and RSAC1 returns immediately. On the other hand, if the number of points in the file was larger than the maximum size of the passed array, RSAC1 takes corrective action by reading only the first MAX points from the file and warns you by setting NERR to -803. (This is why you need to pass the maximum array size to RSAC1.) In this example, we are not concerned about warning conditions, so we only check for a positive NERR. You will have to decide how you want to handle the various error conditions that can occur in your own programs.

RSAC1 C —

```
#include <stdio.h> #include <stdlib.h> #define MAX 1000
main() { float yarray[ MAX ] , beg , del ; int nlen , nerr , max = MAX
; char kname[ 11 ] ;
```

```

    strcpy ( kname , "FILE1" ) ; rsac1( kname, yarray, &nlen, &beg, &del,
&max, &nerr, strlen( kname ) ) ; if ( nerr != 0 ) exit ( nerr ) ; doit ( yarray
, nlen ) ; }

```

Notice that in the call to rsac1, there is an extra parameter after nerr. This is the string length specifier which specifies the length of the string kname. The length of the string does not include a null terminator. Note also that all of the parameters are passed by reference except the string length specifier.

RSAC2 FORTRAN —

```

PROGRAM RSAC PARAMETER (MAX=3000) DIMENSION XAR-
RAY(MAX), YARRAY(MAX) CHARACTER*10 KNAME KNAME='FILE2'
CALL RSAC2(KNAME,YARRAY,NLEN,XARRAY,MAX,NERR) IF(NERR.GT.0)GO
TO 8888 CALL DOMORE(XARRAY,YARRAY,NLEN) 8888 CONTINUE
END

```

There are only a few changes from the first example. Line 3 defines two arrays, instead of one. In line 6 there is one less argument to RSAC2 than RSAC1, the begin time and sampling interval being replaced by the x array.

RSAC2 C —

```

#include <stdio.h> #include <stdlib.h> #define MAX 3000
main() { float xarray[ MAX ] , yarray[ MAX ] , beg , del ; int nlen , nerr
, max = MAX ; char kname[ 11 ] ;
    strcpy ( kname , "FILE2" ) ; rsac2( kname , yarray , &nlen , xarray ,
&max , &nerr , strlen( kname ) ) ; if ( nerr != 0 ) exit ( nerr ) ; domore (
xarray , yarray , nlen ) ; }

```

Header Access Functions FORTRAN —————

```

PROGRAM RSAC PARAMETER (MAX=1000) DIMENSION YARRAY(MAX)
CHARACTER*10 KNAME KNAME='FILE1' CALL RSAC1(KNAME,YARRAY,NLEN,BEG,DEL,M
IF(NERR.GT.0)GO TO 8888 CALL GETFHV('DELTA',DELTA,NERR)
IF(NERR.GT.0)GO TO 8888 CALL GETFHV('B',B,NERR) IF(NERR.GT.0)GO
TO 8888 CALL GETFHV('T1',T1,NERR) IF(NERR.GT.0)GO TO 8888
CALL GETFHV('T2',T2,NERR) IF(NERR.GT.0)GO TO 8888 N1=INT((T1-
B)/DELTA) N2=INT((T2-B)/DELTA) CALL DOPART(YARRAY(N1),N2-
N1+1) 8888 CONTINUE END

```

After the call to RSAC1 in line 6, the header variables for FILE1 will be stored in a common block. The calls to GETFHV in lines 8 through 15 get the four floating point header variables, DELTA, B, T1, and T2. There are two error numbers that can come back from these calls. Error 1336 means that the header variable name is legal but not defined for this file. Error 1337 means that a header variable with that name does not exist. Lines 16 and 17 use the variables from these calls to compute a small piece of the data to be passed to DOPART in line 18 for processing. Other subroutines used to access the header are GETIHV, GETKHV, GETLHV, and GETNHV. These routines are described above and in the appendix. The section on SAC data file format lists header variables by type.

Header Access Functions C ————— #include <stdio.h> #include <stdlib.h> #define MAX 1000

```
main() { long int max = MAX , nlen , nerr ; int N1 , N2 ; float yarray[
MAX ] , beg , del , delta , B , T1 , T2 ; char kname[ 11 ] ;
strcpy ( kname , "FILE1" ) ; rsac1( kname, yarray, &nlen, &beg, &del,
&max, &nerr, strlen( kname ) ) ; if ( nerr != 0 ) exit ( nerr ) ; getfhv (
"DELTA" , &delta , &nerr , 5 ) ; if( nerr != 0 ) exit ( nerr ) ; getfhv ( "B" ,
&B , &nerr , 1 ) ; if ( nerr != 0 ) exit ( nerr ) ; getfhv ( "T1" , &T1 , &nerr ,
2 ) ; if ( nerr != 0 ) exit ( nerr ) ; getfhv ( "T2" , &T2 , &nerr , 2 ) ; if ( nerr
!= 0 ) exit ( nerr ) ; N1 = (int) ( ( ( T1 - B ) / delta ) + 0.5 ) ; N2 = (int) (
( ( T2 - B ) / delta ) + 0.5 ) ; dopart ( yarray + N1 , N2 - N1 + 1 ) ; }
```

WSAC1 FORTRAN —

```
PROGRAM WSAC PARAMETER (MAX=200) DIMENSION YFUNC(MAX)
CHARACTER*10 KNAME KNAME='EXPDATA' BEG=0. DEL=0.02
X=BEG DO 1000 J=1,MAX YFUNC(J)=EXP(-X) 1000 X=X+DEL CALL
WSAC1(KNAME,YFUNC,MAX,BEG,DEL,NERR) END
```

Lines 4 and 5 define the name of the file to be written and like RSAC1 and RSAC2 this name must be a blank filled character variable. Lines 6 through 11 define and create the function. Line 12 calls WSAC1 which writes the data to disk. A minimum header is created, containing only those variables needed to be able to read the file: B, E, DELTA, LEVEN, and NPTS. The calling sequence for WSAC1 is very similar but not identical to that for

RSAC1.

```
WSAC1 C — #include <stdio.h> #include <stdlib.h> #include <math.h>
#define MAX 200
```

```
main() { int max = MAX , j , nerr ; float yfunc[ MAX ] , x , beg = 0. ,
del = 0.02 ; char kname[ 10 ] ;
strcpy ( kname , “EXPDATA” ) ; for ( j = 0 , x = beg ; j < MAX ; j++ ,
x += del ) yfunc[ j ] = exp ( -x ) ; wsac1 ( kname , yfunc , &max , &beg
, &del , &nerr , strlen ( kname ) ) ; }
```

WSAC2 FORTRAN —

```
PROGRAM WSAC PARAMETER (MAX=300) DIMENSION XDATA(MAX),
YDATA(MAX) CHARACTER*10 KNAME KNAME='XYDATA' CONA=12.43
CONB=0.02 CALL GENDAT(CONA,CONB,MAX,XDATA,YDATA,NLEN)
CALL WSAC2(KNAME,YDATA,NLEN,XDATA,NERR) END
```

In lines 6 through 8, x and y data is generated by a function called GENDAT based upon the values of two constants. This data is written to a file called XYDATA by WSAC2 in line 9. Again, WSAC2 writes only a minimum header.

```
WSAC2 C — #include <stdio.h> #include <stdlib.h> #define MAX 300
main() { float xdata[ MAX ] , ydata[ MAX ] ; float cona = 12.43 , conb
= 0.02 ; int max = MAX , nlen , nerr ; char kname[ 11 ] ;
strcpy ( kname , “XYDATA” ) ; gendat ( cona , conb , max , xdata ,
ydata , &nlen ) ; wsac2 ( kname , ydata , &nlen , xdata , &nerr , strlen (
kname ) ) ; }
```

WSAC0 —

To create a SAC data file with more information in the header than WSAC1 and WSAC2 allow, you need to use a set of subroutines that store header variables and then use WSAC0. Below are three examples, the first is similar to the example for WSAC2.

First Example of WSAC0 - Unevenly Spaced Data FORTRAN —

```
PROGRAM WSAC PARAMETER (MAX=300) DIMENSION XDATA(MAX),
YDATA(MAX) CHARACTER*10 KNAME KNAME='XYDATA' CONA=12.43
CONB=0.02 CALL GENDAT(CONA,CONB,MAX,XDATA,YDATA,NLEN)
```

```

CALL NEWHDR CALL SETNHV('NPTS',MAX,NERR) CALL SETLHV('LEVEN',.FALSE.,NERR)
CALL SETFHV('B',XDATA(1),NERR) CALL SETFHV('E',XDATA(MAX),NERR)
CALL SETIHV('IFTYPE','IXY',NERR) CALL SETFHV('USER0',CONA,NERR)
CALL SETFHV('USER1',CONB,NERR) CALL SETKHV('KUSER0','GENDAT',NERR)
CALL WSAC0(KNAME,XDATA,YDATA,NERR) 8888 CONTINUE END

```

This example is the same as the one above through line 8. The call to NEWHDR in line 9 sets all header fields to their default values. Omitting the call to NEWHDR will retain any header fields from a previous call to RSAC0 or RSAC1; allowing you to read a SAC file, manipulate the data, and write the file out with all the original header contents. If you are creating more than one file in a program, you can use NEWHDR to reset the header to its default state for each file. Lines 10 through 17 define some header values for this file. All of the header variables, except the auxiliary ones (denoted by an A in the appendix) may be set using these subroutine calls. Note the use of USER0, USER1, and KUSER0 to store information unique to this application. No error checking is done after these calls because we know we are using the names of legitimate SAC header variables. The data file with this header is then written by calling WSAC0 in line 18. Note that the calling sequence is significantly different for WSAC0 than for WSAC1 and WSAC2. If the data had been evenly spaced, the x array could be a dummy array.

First Example of WSAC0 C ————— #include <stdio.h> #include <stdlib.h> #define MAX 300 #define FALSE 0 #define TRUE 1

```

main() { int max = MAX , nlen , nerr ; float xdata[ MAX ] , ydata[
MAX ] ; float cona = 12.43, conb = 0.02 ; char kname[ 11 ] ; long false =
FALSE ;

```

```

    strcpy ( kname , "XYDATA" ) ; gendat ( cona , conb , max , xdata ,
ydata , &nlen ) ; newhdr ( ) ; setnhv ( "npts" , &max , &nerr , 4 ) ; setlhv
( "leven" , &>false , &nerr , 5 ) ; setfhv ( "b" , xdata , &nerr , 1 ) ; setfhv (
"e" , xdata + max , &nerr , 1 ) ; setihv ( "iftype" , "ixy" , &nerr , 6 , 3 ) ;
setfhv ( "user0" , &cona , &nerr , 5 ) ; setfhv ( "user1" , &conb , &nerr , 5
) ; setkhv ( "kuser0" , "gendat" , &nerr , 6 , 6 ) ; wsac0 ( kname , xdata ,
ydata , &nerr , strlen ( kname ) ) ; }

```

Notice that setihv and setkhv each have two character strings in their parameter lists. These two functions also each have two string length specifiers at the end of the parameter lists.

Second Example of WSAC0 - XYZ (3-D) files FORTRAN —————

```

PROGRAM WSAC PARAMETER (MAX=36) DIMENSION DUMMY(MAX),
ZDATA(MAX) CHARACTER*10 KNAME KNAME='XYZDATA' CALL
GENDAT(MAX,YDATA,NLEN) CALL NEWHDR CALL SETNHV('NPTS',MAX,NERR)
CALL SETLHV('LEVEN',.TRUE.,NERR) CALL SETIHV('IFTYPE','IXYZ',NERR)
CALL SETNHV('NXSIZE',6,NERR) CALL SETNHV('NYSIZE',6,NERR)
CALL SETFHV('XMINIMUM', minimum,NERR) CALL SETFHV('XMAXIMUM',
maximum,NERR) CALL SETFHV('YMINIMUM', minimum,NERR) CALL
SETFHV('YMAXIMUM', maximum,NERR) CALL WSAC0(KNAME,DUMMY,ZDATA,NERR)
END

```

Although data in SAC memory is stored in a linear 1-D array, think of the Z data as being placed in a 2-D grid, in the order left-to-right, bottom-to-top. See the CONTOUR command for additional information.

Second Example of WSAC0 C ————— #include <stdio.h> #include <stdlib.h> #define MAX 36 #define FALSE 0 #define TRUE 1

```

main() { int max = MAX , nlen , nerr , six = 6 ; float dummy[ MAX
] , zdata[ MAX ] ; float Xminimum = 7e+05 , Xmaximum = 8.25e+07 ,
Yminimum = 0.0 , Ymaximum = 2.06e+11 ; char kname[ 11 ] ; long true =
TRUE ;

```

```

strcpy ( kname , "XYZDATA" ) ; gendat ( max , zdata , &nlen ) ; newhdr
() ; setnhv ( "NPTS" , &max , &nerr , 4 ) ; setlhv ( "LEVEN" , &true , &nerr
, 5 ) ; setihv ( "IFTYPE" , "IXYZ" , &nerr , 6 , 4 ) ; setnhv ( "NXSIZE"
, &six , &nerr , 6 ) ; setnhv ( "NYSIZE" , &six , &nerr , 6 ) ; setfhv (
"XMINIMUM" , &Xminimum , &nerr , 8 ) ; setfhv ( "XMAXIMUM" ,
&Xmaximum , &nerr , 8 ) ; setfhv ( "YMINIMUM" , &Yminimum , &nerr
, 8 ) ; setfhv ( "YMAXIMUM" , &Ymaximum , &nerr , 8 ) ; wsac0 ( kname
, dummy , zdata , &nerr , strlen ( kname ) ) ; }

```

Note that for setnhv, the second parameter is the address of a variable assigned the value 6 instead of just the value 6. This is because all parame-

ters are passed by reference except string length specifiers.

Third Example of WSAC0 The explanation precedes the code. ———

The following example converts data from a fictitious seismic network to SAC data files. This network contains seven single component stations and one three component station. Lines 2 through 6 set up storage for the seismic data and define the names of the files. Lines 7 through 13 initialize the header and set header variables that are independent of the data being converted. Lines 14 through 16 query the user for an event identification number and then extracts the data from the data base using GETDAT. In this example, NDATA is the number of points per component for this event, TIME is the epochal time of the beginning of this event, and ELAT and ELON are the event latitude and longitude. Lines 17 and 18 convert this time to the SAC zero time and lines 19 through 25 define some more SAC header fields. The loop in lines 26 through 33 define the station coordinates and create SAC data files for the 8 vertical components. Lines 34 through 38 create the data files for the two horizontal components.

Third Example of WSAC0 FORTRAN ———

```

PROGRAM WSACH PARAMETER (MCOMP=9, MDATA=4000) DI-
MENSION SDATA(MDATA,MCOMP+1) CHARACTER*10 KSTNM CHAR-
ACTER KNAME(MCOMP+1)*10 DATA KNAME/'STAZ','STBZ','STCZ','STDZ','STEZ',
# 'STFZ','STGZ','STHZ','STHN','STHE' / CALL NEWHDR CALL SETIHV('IFTYPE','ITIME',NER
CALL SETIHV('IDEP','IVEL',NERR) CALL SETIHV('IZTYPE','IB',NERR)
CALL SETFHV('B',0.,NERR) CALL SETLHV('LEVEN',.TRUE.,NERR)
CALL SETFHV('DELTA',0.025,NERR) PRINT *, 'Enter event id' READ(*,*)IEVID
CALL GETDAT(IEVID,SDATA,NDATA,TIME,ELAT,ELON) CALL CN-
VTIM(TIME,NZYEAR,NZJDAY,NZHOURL, # NZMIN,NZSEC,NZMSEC)
CALL SETNHV('NPTS',NDATA,NERR) CALL SETFHV('EVLA',ELAT,NERR)
CALL SETFHV('EVLO',ELON,NERR) WRITE(KEVNM,'(I5)')IEVID CALL
SETKHV('KEVNM',KEVNM,NERR) CALL SETFHV('CMPAZ',0.,NERR)
CALL SETFHV('CMPINC',0.,NERR) DO 2000 J=1,8 KSTNM=KNAME(J)(1:3)
CALL SETKHV('KSTNM',KSTNM,NERR) CALL GETLOC(KSTNM,STLA,STLO)
CALL SETFHV('STLA',STLA,NERR) CALL SETFHV('STLO',STLO,NERR)

```

```

CALL WSAC0(KNAME(J),XDUMMY,SDATA(1,J),NERR) 2000 if(NERR.NE.0)GO
TO 9000 CALL SETFHV('CMPINC',90.,NERR) CALL WSAC0(KNAME(9),XDUMMY,SDATA(1,9),
IF(NERR.NE.0)GO TO 9000 CALL SETFHV('CMPAZ',90.,NERR) CALL
WSAC0(KNAME(10),XDUMMY,SDATA(1,10),NERR) 9000 CONTINUE
END

```

Third Example of WSAC0 C —————

```

#include <stdio.h> #include <stdlib.h> #define MCOMP 10 #define MDATA
4000 #define FALSE 0 #define TRUE 1

main() { float sdata[ MCOMP ][ MDATA ] , xdummy[ MDATA ] ; float
zero = 0. , fortieth = 0.025 , ninty = 90. ; float time , elat , elon , stla , stlo
; char kname[ MCOMP ][ 11 ] = { "STAZ" , "STBZ" , "STCZ" , "STDZ" ,
"STEZ" , "STFZ" , "STGZ" , "STHZ" , "STHN" , "STHE" } ; char kevm[
11 ] , * kstnm ; int nerr , ievid , ndata , j ; long true = TRUE , false =
FALSE ;

newhdr ( ) ; setihv ( "IFTYPE" , "ITIME" , &nerr , 6 , 5 ) ; setihv (
"IDEP" , "IVEL" , &nerr , 4 , 4 ) ; setihv ( "IZTYPE" , "IB" , &nerr , 6 ,
2 ) ; setfhv ( "B" , &zero , &nerr , 1 ) ; setlhv ( "LEVEN" , &>true , &nerr
, 5 ) ; setfhv ( "DELTA" , &fortieth , &nerr , 5 ) ;

printf ( "\nEnter event id: " ) ; gets ( kevm ) ; ievid = atoi ( kevm )
; getdat ( ievid , sdata , ndata , time , elat , elon ) ;

setnhv ( "NPTS" , ndata , &nerr , 4 ) ; setfhv ( "EVLA" , elat , &nerr
, 4 ) ; setfhv ( "EVLO" , elon , &nerr , 4 ) ; setkhv ( "KEVM" , kevm ,
&nerr , 5 ) ; setfhv ( "CMPAZ" , &zero , &nerr , 5 ) ; setfhv ( "CMPINC"
, &zero , &nerr , 6 ) ;

for ( j = 0 ; j < MCOMP - 2 ; j++ ) { kstnm = kname[ j ] ; setkhv (
"KSTNM" , kstnm , &nerr , 5 , strlen ( kstnm ) ) ; getloc ( kstnm , &stla ,
&stlo ) ; setfhv ( "STLA" , &stla , &nerr , 4 ) ; setfhv ( "STLO" , &stlo ,
&nerr , 4 ) ; wsac0 ( kstnm , xdummy , sdata[ j ] , &nerr , strlen ( kstnm )
) ; }

setfhv ( "CMPINC" , &ninty , &nerr , 6 ) ; wsac0( kname[ 9 ] , xdummy,
sdata[ 9 ] , &nerr, strlen( kname[ 9 ] ) ) ;

setfhv ( "CMPAZ" , &ninty , &nerr , 5 ) ; wsac0( kname[ 10 ] , xdummy,
sdata[ 10 ] , &nerr, strlen( kname[ 10 ] ) ) ; }

```

2.8.1 SAC Data File Format

SAC2000 User's Manual

SAC Data File Format

Overview This section discusses the contents of the SAC data file, describes the binary and alphanumeric formats of this file, and documents the SAC header in detail.

Contents Each signal is stored on disk in a separate SAC data file. These files contain a fixed length header section followed by one or two data sections. The header contains floating point, integer, logical, and character fields. Evenly spaced data files have only one data section which contains the dependent variable. Unevenly spaced data and spectral data files contain two data sections. For unevenly spaced data, the first data section contains the dependent variable and the second contains the independent variable. For spectral files the first component is either the amplitude or the real component and the second component is either the phase or imaginary component.

Binary Format This is the format that you will use most often. It is used in SAC itself(READ and WRITE commands) and in the subroutine library (RSAC1, RSAC2, WSAC1, WSAC2, WSAC0.) These are binary (unformatted) files so that they can be quickly read from disk into memory. The header is 158 32-bit words in length, followed by the data section(s). In order to rapidly read only a small section of a data file (see the CUT command), these files also have a physical record length of 512 bytes (128 32-bit words) and are opened for direct-access. There is no physical record structure. This format is shown schematically in the following figure.

Structure of SAC Binary Data File —————

HEADER SECTION FIRST DATA SECTION SECOND DATA SECTION* ————— - start word: 0 start word: 158 start word: 158+NPTS word length: 158 word length: NPTS word length: NPTS contents: see table contents: contents: - dependent variable - independent variable - amplitude if evenly spaced - real component - phase - imaginary component

* if present

Binary Header The following table shows the contents and layout of the SAC binary data file header. The W and T columns give the beginning word and header data type for the header variables named on that line. These header variables and data types are described later in this section. If the name is INTERNAL then that variable is internal to SAC and not normally of interest to the user. If the name is UNUSED then that variable is not currently being used. For any given file, some of these variables will not have meaningful values. These are referred to as “undefined variables” for that file. For each data type, a special value signifies this undefined state. They are listed in a table at the end of this section.

W T NAMES - - - - -

```

0 F DELTA DEPMIN DEPMAX SCALE ODELTA
5 F B E O A INTERNAL
10 F T0 T1 T2 T3 T4
15 F T5 T6 T7 T8 T9
20 F F RESP0 RESP1 RESP2 RESP3
25 F RESP4 RESP5 RESP6 RESP7 RESP8
30 F RESP9 STLA STLO STEL STDP
35 F EVLA EVLO EVEL EVDP MAG
40 F USER0 USER1 USER2 USER3 USER4
45 F USER5 USER6 USER7 USER8 USER9
50 F DIST AZ BAZ GCARC INTERNAL
55 F INTERNAL DEPMEN CMPAZ CMPINC XMINIMUM
60 F XMAXIMUM YMINIMUM YMAXIMUM UNUSED UNUSED
65 F UNUSED UNUSED UNUSED UNUSED UNUSED
70 I NZYEAR NZJDAY NZHOUR NZMIN NZSEC
75 I NZMSEC NVHDR NORID NEVID NPTS
80 I INTERNAL NWFID NXSIZE NYSIZE UNUSED
85 I IFTYPE IDEP IZTYPE UNUSED IINST
90 I ISTREG IEVREG IEVTYP IQUAL ISYNTH
95 I IMAGTYP IMAGSRC UNUSED UNUSED UNUSED
100 I UNUSED UNUSED UNUSED UNUSED UNUSED

```

105 L LEVEN LPSPOL LOVROK LALDA UNUSED

110 K KSTNM KEVNM*

116 K K HOLE KO KA

122 K KT0 KT1 KT2

128 K KT3 KT4 KT5

134 K KT6 KT7 KT8

140 K KT9 KF KUSER0

146 K KUSER1 KUSER2 KCMPTM

152 K KNETWK KDATRD KINST

KEVNM is 16 characters (4 words) long.

All other K fields are 8 characters (2 words) long.

Alphanumeric Format

This file is essentially the alphanumeric equivalent of the SAC binary data file. The header section is stored on the first 30 cards. This is followed by one or two data sections. The data is in 5G15.7 format. The following table shows the card number, formats and names of the variables on the header section cards.

CN FORMAT NAMES — — — — —

01 (5G15.7) DELTA DEPMIN DEPMAX SCALE ODELTA

02 (5G15.7) B E O A INTERNAL

03 (5G15.7) T0 T1 T2 T3 T4

04 (5G15.7) T5 T6 T7 T8 T9

05 (5G15.7) F RESP0 RESP1 RESP2 RESP3

06 (5G15.7) RESP4 RESP5 RESP6 RESP7 RESP8

07 (5G15.7) RESP9 STLA STLO STEL STDP

08 (5G15.7) EVLA EVLO EVEL EVDP MAG

09 (5G15.7) USER0 USER1 USER2 USER3 USER4

10 (5G15.7) USER5 USER6 USER7 USER8 USER9

11 (5G15.7) DIST AZ BAZ GCARC INTERNAL

12 (5G15.7) INTERNAL DEPMEN CMPAZ CMPINC XMINIMUM

13 (5G15.7) XMAXIMUM YMINIMUM YMAXIMUM UNUSED UN-
USED

14 (5G15.7) UNUSED UNUSED UNUSED UNUSED UNUSED

15 (5I10) NZYEAR NZJDAY NZHOUR NZMIN NZSEC
 16 (5I10) NZMSEC NVHDR NORID NEVID NPTS
 17 (5I10) INTERNAL NWDID NXSIZE NYSIZE UNUSED
 18 (5I10) IFTYPE IDEP IZTYPE UNUSED IINST
 19 (5I10) ISTREG IEVREG IEVTYP IQUAL ISYNTH
 20 (5I10) IMAGTYP IMAGSRC UNUSED UNUSED UNUSED
 21 (5I10) UNUSED UNUSED UNUSED UNUSED UNUSED
 22 (5I10) LEVEN LPSPOL LOVROK LCALDA UNUSED
 23 (A8,A16) KSTNM KEVNM
 24 (3A8) KHOLE KO KA
 25 (3A8) KT0 KT1 KT2
 26 (3A8) KT3 KT4 KT5
 27 (3A8) KT6 KT7 KT8
 28 (3A8) KT9 KF KUSER0
 29 (3A8) KUSER1 KUSER2 KCMPNM
 30 (3A8) KNETWK KDATRD KINST

Sample Alphanumeric Data File

The header section and first five lines of the data section of a sample SAC alphanumeric data file is shown below. You can reproduce this file (with the entire data section) on your system by executing the following commands:

u: FUNCGEN SEISMOGRAM u: WRITE ALPHA TEMP1

You can then convert this alphanumeric file to a binary one and read it into SAC with the following commands:

u: CONVERT FROM ALPHA TEMP1 TO SAC TEMP2 u: READ TEMP2

This little test shows the equivalence of the alphanumeric and binary file formats.

0.1000000e-01 -1.569280 1.520640 -12345.00 -12345.00 9.459999 19.45000
 0. 10.47000 2.000000 -12345.00 20.00000 -12345.00 -12345.00 -12345.00 -
 12345.00 -12345.00 -12345.00 -12345.00 -12345.00 17.78000 -12345.00 -12345.00
 -12345.00 -12345.00 -12345.00 -12345.00 -12345.00 -12345.00 -12345.00 -
 12345.00 87.99997 -120.0000 -12345.00 -12345.00 47.99997 -125.0000 -12345.00

```

-12345.00 -12345.00 123.4560 -12345.00 -12345.00 -12345.00 -12345.00 -12345.00
-12345.00 -12345.00 -12345.00 -12345.00 4461.052 0.2718981 185.2046 40.18594
-12345.00 -12345.00 -0.9854718e-01 0. 0. -12345.00 -12345.00 -12345.00
-12345.00 -12345.00 -12345.00 -12345.00 -12345.00 -12345.00 -12345.00 -
12345.00 1981 88 10 38 14 0 6 0 0 1000 -12345 -12345 -12345 -12345 -12345
1 50 9 -12345 -12345 -12345 -12345 42 -12345 -12345 -12345 -12345 -12345
-12345 -12345 -12345 -12345 -12345 -12345 -12345 1 1 1 1 0 CDV K8108838
-12345 HOLE IPD0 XYZ -12345 KT1 -12345 -12345 -12345 -12345 -12345
-12345 -12345 -12345 ABKD USER0 -12345 -12345 -12345 -12345 -12345 -
0.9728001e-01 -0.9728001e-01 -0.9856002e-01 -0.9856002e-01 -0.9728001e-01
-0.9600000e-01 -0.9472002e-01 -0.9344001e-01 -0.9344001e-01 -0.9344001e-01
-0.9344001e-01 -0.9344001e-01 -0.9472002e-01 -0.9472002e-01 -0.9344001e-01
-0.9344001e-01 -0.9216000e-01 -0.9216000e-01 -0.9216000e-01 -0.9216000e-01
-0.9088002e-01 -0.9088002e-01 -0.9216000e-01 -0.9344001e-01 -0.9472002e-01

```

TABLE: Header Variables

This table lists the header variables, their types, and descriptions. They are grouped by category: required fields, time fields, phase picks, instrument parameters, station parameters, event parameters, misc. The header types are defined in the second table.

Required Fields

name type description --- ---

NPTS N Number of points per data component. [required]

NVHDR N Header version number. Current value is the integer 6. [required]

B F Beginning value of the independent variable. [required]

E F Ending value of the independent variable. [calculated]

IFTYPE I Type of file [required]:

= ITIME {Time series file}

= IRLIM {Spectral file—real and imaginary}

= IAMPH {Spectral file—amplitude and phase}

= IXY {General x versus y data}

= IXYZ {General XYZ (3-D) file}

LEVEN L TRUE if data is evenly spaced. [required]

DELTA F Increment between evenly spaced samples (nominal value).
[required]
ODELTA F Observed increment if different from nominal value.
IDEP I Type of dependent variable:
= IUNKN (Unknown)
= IDISP (Displacement in nm)
= IVEL (Velocity in nm/sec)
= IVOLTS (Velocity in volts)
= IACC (Acceleration in nm/sec/sec)
SCALE F Multiplying scale factor for dependent variable [not currently
used]
DEPMIN F Minimum value of dependent variable. [calculated]
DEPMAX F Maximum value of dependent variable. [calculated]
DEPMEN F Mean value of dependent variable. [calculated]
Time Fields
name type description --- ---
NZYEAR N GMT year corresponding to reference (zero) time in file.
NZJDAY N GMT julian day.
NZHOUR N GMT hour.
NZMIN N GMT minute.
NZSEC N GMT second.
NZMSEC N GMT millisecond.
NZDTTM N GMT date-time array. Six element array equivalenced to
NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, and NZMSEC.
KZDATE A Alphanumeric form of GMT reference date.
Derived from NZYEAR and NZJDAY.
KZTIME A Alphanumeric form of GMT reference time.
Derived from NZHOUR, NZMIN, NZSEC, and NZMSEC.
IZTYPE I Reference time equivalence:
= IUNKN (Unknown)
= IB (Begin time)
= IDAY (Midnight of refernece GMT day)
= IO (Event origin time)

= IA (First arrival time)
 = ITn (User defined time pick n, n=0,9)
 O F Event origin time (seconds relative to reference time.)
 KO A Event origin time identification.
 Phase Picks
 name type description --- ---
 A F First arrival time (seconds relative to reference time.)
 KA K First arrival time identification.
 F F Fini or end of event time (seconds relative to reference time.)
 KF K Fini identification.
 Tn F User defined time picks or markers, n=0,9.
 (seconds relative to reference time.)
 KTn K User defined time pick identifications, n=0,9.
 Instrument Fields
 name type description --- ---
 KINST K Generic name of recording instrument.
 IINST I Type of recording instrument. [not currently used]
 RESPn F Instrument response parameters, n=0,9. [not currently used]
 Station Fields
 name type description --- ---
 KNETWK K Name of seismic network.
 KSTNM K Station name.
 ISTREG I Station geographic region. [not currently used]
 STLA F Station latitude (degrees, north positive).
 STLO F Station longitude (degrees, east positive).
 STEL F Station elevation (meters). [not currently used]
 STDP F Station depth below surface (meters). [not currently used]
 CMPAZ F Component azimuth (degrees clockwise from north).
 CMPINC F Component incident angle (degrees from vertical).
 KCMPNM K Component name.
 KSTCMP A Station component. Derived from KSTNM, CMPAZ, and
 CMPINC.

LPSPOL L TRUE if station components have a positive polarity (left-hand rule).

Event Fields

name type description --- -- —————

KEVNM K Event name.

IEVREG I Event geographic region. [not currently used]

EVLA F Event latitude (degrees, north positive).

EVLO F Event longitude (degrees, east positive).

EVEL F Event elevation (meters). [not currently used]

EVDP F Event depth below surface (meters). [not currently used]

MAG F Event magnitude.

IMAGTYP I Magnitude type:

= IMB (Bodywave Magnitude)

= IMS (Surfacewave Magnitude)

= IML (Local Magnitude)

= IMW (Moment Magnitude)

= IMD (Duration Magnitude)

= IMX (User Defined Magnitude)

IMAGSRC I Source of magnitude information:

= INEIC (National Earthquake Information Center)

= IPDE (Preliminary Determination of Epicenter)

= IISC (International Seismological Centre)

= IREB (Reviewed Event Bulletin)

= IUSGS (US Geological Survey)

= IBRK (UC Berkeley)

= ICALTECH (California Institute of Technology)

= ILLNL (Lawrence Livermore National Laboratory)

= IEVLOC (Event Location (computer program))

= IJSOP (Joint Seismic Observation Program)

= IUSER (The individual using SAC2000)

= IUNKNOWN (unknown)

IEVTYP I Type of event:

= IUNKN (Unknown)

- = INUCL (Nuclear event)
- = IPREN (Nuclear pre-shot event)
- = IPOSTN (Nuclear post-shot event)
- = IQUAKE (Earthquake)
- = IPREQ (Foreshock)
- = IPOSTQ (Aftershock)
- = ICHEM (Chemical explosion)
- = IQB0 (Probable mine blast)
- = IQB (Quarry or mine blast confirmed by quarry)
- = IQB1 (Quarry/mine blast with designed shot info-ripple fired)
- = IQB2 (Quarry/mine blast with observed shot info-ripple fired)
- = IQBX (Quarry or mine blast - single shot)
- = IQC (Mine collapse)
- = IQMT (Quarry/mining-induced events: tremors and rockbursts)
- = IEQ0 (Probable earthquake)
- = IEQ (Earthquake)
- = IEQ1 (Earthquakes in a swarm or aftershock sequence)
- = IEQ2 (Felt earthquake)
- = IEQ3 (Damaging earthquake)
- = IME (Marine explosion)
- = IEX (Other explosion)
- = IEX0 (Probable explosion)
- = INU (Nuclear explosion)
- = INC (Nuclear cavity collapse)
- = IGEY (Geyser)
- = ILIT (Light)
- = IMET (Meteoric origin)
- = IODOR (Odors)
- = IO_ (Other source of known origin)
- = IL (Local event of unknown origin)
- = IR (Regional event of unknown origin)
- = IT (Teleseismic event of unknown origin)
- = IU (Undetermined or conflicting information)

= IOTHER (Other)
 NEVID N Event ID (CSS 3.0)
 NORID N Origin ID (CSS 3.0)
 NWFID N Waveform ID (CSS 3.0)
 KHOLE K Hole identification if nuclear event.
 DIST F Station to event distance (km).
 AZ F Event to station azimuth (degrees).
 BAZ F Station to event azimuth (degrees).
 GCARC F Station to event great circle arc length (degrees).

Note: calculations of DIST, AZ, BAZ, and GCARC are based upon the reference spheroid of 1968 and are defined by the major radius (RAD) and the flattening (FL). DIST is computed by Rudoe's formula given in GEODESY, section 2.15(b).

Miscellaneous Fields

name type description --- -- —————

LCALDA L TRUE if DIST, AZ, BAZ, and GCARC are to be calculated from station and event coordinates.

IQUAL I Quality of data [not currently used]:

= IGOOD (Good data)
 = IGLCH (Glitches)
 = IDROP (Dropouts)
 = ILOWSN (Low signal to noise ratio)
 = IOTHER (Other)

ISYNTH I Synthetic data flag [not currently used]:

= IRLDTA (Real data)
 = ????? (Flags for various synthetic seismogram codes)

KDATRD K Date data was read onto computer.

USERn F User defined variable storage area, n=0,9.

KUSERn K User defined variable storage area, n=0,2.

LOVROK L TRUE if it is okay to overwrite this file on disk.

NXSIZE N Spectral Length (Spectral files only)

NYSIZE N Spectral Width (Spectral files only)

XMINIMUM F Minimum value of X (Spectral files only)

XMAXIMUM F Maximum value of X (Spectral files only)

YMINIMUM F Minimum value of Y (Spectral files only)

YMAXIMUM F Maximum value of Y (Spectral files only)

TABLE: Header Data Types

This table lists the header types and their definitions. The third column lists the special value used to signify that a particular header variable is undefined in a particular file.

type	definition	undefined	description
------	------------	-----------	-------------

F	Floating	-12345.0	Single precision.
---	----------	----------	-------------------

N	Integer	-12345	Name begins with an "N".
---	---------	--------	--------------------------

I	Enumerated	-12345	Name begins with an "I". Has a limited set of integer values. Each value is given a specific name. Each value represents a specific condition. Functions use the equivalent alphanumeric name.
---	------------	--------	--

L	Logical	FALSE	Name begins with an "L". Value is either TRUE or FALSE.
---	---------	-------	---

K	Alphanumeric	"-12345"	Name begins with a "K". Either 8 or 16 characters long.
---	--------------	----------	---

A	Auxiliary	Not really in the header.	Derived from other header fields.
---	-----------	---------------------------	-----------------------------------

Enumerated Header

Enumerated Header Field Values

The enumerated header field values are stored in the header as integers. Their names and values are given in the table below.

name	value	name	value
------	-------	------	-------

itime	01	irlim	02
-------	----	-------	----

iamph	03	ixy	04
-------	----	-----	----

iunkn	05	idisp	06
-------	----	-------	----

ivel	07	iacc	08
------	----	------	----

ib	09	iday	10
----	----	------	----

io	11	ia	12
----	----	----	----

it0	13	it1	14
-----	----	-----	----

it2	15	it3	16
-----	----	-----	----

it4	17	it5	18
-----	----	-----	----

it6	19	it7	20
-----	----	-----	----

it8	21	it9	22
-----	----	-----	----

iradnv 23 itannv 24
 iradev 25 itanev 26
 inorth 27 ieast 28
 ihorza 29 idown 30
 iup 31 illlbb 32
 iwwsn1 33 iwwsn2 34
 ihglp 35 isro 36
 inucl 37 ipren 38
 ipostn 39 iquake 40
 ipreq 41 ipostq 42
 ichem 43 iother 44
 igood 45 iglch 46
 idrop 47 ilowsn 48
 irltda 49 ivolts 50
 ixyz 51 imb 52
 ims 53 iml 54
 imw 55 imd 56
 imx 57 ineic 58
 ipde 59 iisc 60
 ireb 61 iusgs 62
 ibrk 63 icaltech 64
 illnl 65 ievloc 66
 ijsop 67 iuser 68
 iunknown 69 iqb 70
 iqb1 71 iqb2 72
 iqbz 73 iqmt 74
 ieq 75 ieq1 76
 ieq2 77 ime 78
 iex 79 inu 80
 inc 81 io_ 82
 il 83 ir 84
 it 85 iu 86

2.8.2 API

Application Programmer Interface

*** How to use SAC2000's external function capability. ***

Summary: Build a shared object according to specifications use load command to load into sac run the command as desired

STEP 1: compile srcs into one or more external libs. For example the following loads a command called flipxy into a shared object library libext.so

```
cc -o /us/peterg/lib/libext.so -G flipxy.c
```

By default sac will call the command flipxy. Note: we can have multiple srcs on this line. e.g., cc -o /us/peterg/lib/libext.so -G extern.c velarr.c flipxy.c See the load command for more details.

STEP 2: Setup the SACSOLIST and LD_LIBRARY_PATH environments.

setenv SACSOLIST "Shared library name" e.g., setenv SACSOLIST libext.so Use space delim and quotes for multiple libs

setenv LD_LIBRARY_PATH \${LD_LIBRARY_PATH}:/Path_to_Shared_Library e.g., setenv LD_LIBRARY_PATH \${LD_LIBRARY_PATH}:/us/peterg/lib

STEP 3: {Optional} Use init script to load automatically. alias sac "/usr/local/bin/sac /us/peterg/bin/sacmac/sac.init" where sac.init includes the users startup commands including load external_function_command_name The following is an example initialization file: setmacro /us/peterg/bin/sacmac window 1 x .34 1.0 y 0.5 1.0 qdp 5000 load flipxy

STEP 4: Startup sac and load the command (if you didn't load it via STEP 3). e.g., SAC_i load flipxy SAC_i fg SAC_i flipxy

*** How to Share with other users: ***

STEP 1: Give other users your function or shared library and this notes file. Tell the other users to do step 2-4.

STEP 2: You can also send external function you would like distributed as part of the SAC2000 distribution to:

peterg@llnl.gov

Please include a brief description with a relevant references that we can include as part of the documentation. All contributions will be acknowledged

appropriately.

SAC Command Reference Manual EXTERNAL COMMAND INTERFACE

SUMMARY: Description of interface for external commands callable by SAC.

DESCRIPTION:

C language interface

The following definitions and structures will be used to pass data into and out of external functions specified by the user. These external commands will be loaded by SAC at run time by executing the LOAD command (See LOAD help page for further details).

The application programming interface for external functions (commands) is:

```
long ext_func(argc, argv, call_data, update) int argc; char **argv; sac_files
*call_data; long *update;
```

This function should return a long to be used as an error status flag. By convention, if this function returns a non-zero value, SAC will indicate that an error occurred within this function. By default, SAC will print out the error number returned. If the user wants to add a customized error message, this can be done by editing the messages file in the SAC aux directory. Care must be taken not to use an error number that has already been used in another context.

Where argc and argv contain the command line arguments, defined the same as the command line arguments for a C program. argc is set to the number of arguments, and argv contains the tokenized command line. argc is always greater than or equal to one, since argv[0] contains the command name.

sac_files is a pointer to a call_data struct which is used to package the sac headers and data for efficient communication with the external function. This data structure is defined in the file extfunc.h, which must be included in the external function.

update is a flag which tells SAC how to handle the data returned from the external function. It should be set to one of the enumerated values:

APPEND, REPLACE, or IGNORE. If this flag is set to APPEND, the data returned from the external function will be appended to the existing data file list (files in memory). If set to REPLACE, the returned data will replace the data in memory (which is the way that most SAC commands work). If set to IGNORE, it will be disregarded.

Several support routines are provided to facilitate header access. They include:

`sac_header *makehdr(sac_header *header_in)` Allocate a new header struct. If `header_in` is not NULL, copy its values. If `header_in` is NULL, initialize the new header to default values.

`long getehdr(sac_header *header, char *fieldname, long *error)` Return the value of the enumerated header field pointed to by `fieldname` from the header struct pointed to by `header`.

`void setehdr(sac_header *header, char *fieldname, long value, long *error)` Set the enumerated field specified in `fieldname` to value in the header specified in `header`.

`float getfhdr(sac_header *header, char *fieldname, long *error)` Return the value of the floating point header field `fieldname` from the header specified in `header`.

`void setfhdr(sac_header *header, char *fieldname, float value, long *error)` Set the floating point field `fieldname` to value in header pointed to by `header`.

`long getnhdr(sac_header *header, char *fieldname, long *error)` Return the value of the long field specified in `fieldname` from the header specified by `header`.

`void setnhdr(sac_header *header, char *fieldname, long value, long *error)` Set the long header field `fieldname` to value in the header specified by `header`.

`long getlhdr(sac_header *header, char *fieldname, long *error)` Return the value of the logical header field `fieldname` from the header specified by `header`.

`void setlhdr(sac_header *header, char *fieldname, long value, long *error)` Set the logical header field `fieldname` to value in the header `header`.

char *getahdr(sac_header *header, char *fieldname, long *error) Return a pointer to the value of the character header field fieldname from the header specified by header. This function returns a pointer to the actual header field. You should not modify this and also should not free this returned address.

void setahdr(sac_header *header, char *fieldname, char *value, long *error) Set the character header field fieldname to the value pointed to by value in the header specified by header.

All header access routines return zero in the error variable if no error occurred, otherwise they return non-zero.

The file extfunc.h contains tables of the names of the various header fields which can be returned or set by the above functions. It also contains definitions of all the enumerated values known to SAC.

FORTRAN language interface

The FORTRAN language interface to external commands consists of a C language function which maps data into and out of a FORTRAN routine having the following interface:

```
subroutine fmycommand(fargs, fyinput, fxinput, numfiles, nptsmax, fer-
ror)
include 'fext_params'
character*(*) fargs real*4 fyinput(nptsmax,numfiles) real*4 fxinput(nptsmax,numfiles)
integer*4 numfiles, nptsmax, ferror
```

Where fargs is the command line, blank delimited. fyinput contains the input y data. fyinput is zero filled for data consisting of less than nptsmax points. fxinput contains the x data for unevenly spaced files. In the case of evenly spaced data, fxinput is all zeroes. numfiles is the number of input files, nptsmax is the maximum number of points of all the input files and ferror is an error return flag.

The include file "fext_params" contains parameters defining the valid enumerated header values.

The C language function referred to above is fextern.c.template.

Several support routines are provided to facilitate header access. They include:

fgetahdr(integer*4 hdr_index, character fieldname, character value, integer*4 error) Returns value of character header field fieldname in value. Value returned is from header(hdr_index), where hdr_index ranges from 1 to numfiles.

fsetahdr(integer*4 hdr_index, character fieldname, character value, integer*4 error) Set the value of character header field fieldname to value.

fgetehdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error) Returns the value of enumerated header field fieldname in value.

fsetehdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error) Set the value of enumerated header field fieldname to value.

fgetfhdr(integer*4 hdr_index, character fieldname, real*4 value, integer*4 error) Returns the value of real header field fieldname in value.

fsetfhdr(integer*4 hdr_index, character fieldname, real*4 value, integer*4 error) Set the value of real header field fieldname to value.

fgetlhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error) Returns the value of logical header field fieldname in value.

fsetlhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error) Set the value of logical header field fieldname to value.

fgetnhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error) Returns the value of integer header field fieldname in value.

fsetnhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error) Set the value of integer header field fieldname to value.

2.8.3 API Howto

2.8.4 Subroutine Calls

SAC2000 User's Manual

Appendix: Subroutine Calls

Overview This section documents the SAC subroutines discussed in this manual. They are sorted alphabetically by name and include the FORTRAN calling sequence, C prototype, purpose, input arguments, and output arguments. The following notation is used in describing the arguments:

f Floating point (real) variable. i Integer variable, long integer (32 bits.)
c Character variable, arbitrary length, blank padded. c n Character variable,
n characters long. k Character variable, arbitrary length, blank padded. l
Logical variable (declared as long in C). xa An array of the type x, where x
is one of the above.

For C functions, parameter names ending with `_s` are string length spec-
ifiers. Output strings should be padded with spaces and null terminated
in the last element of the array. Input strings can be null terminated at
the end of the meaningful characters, or can be padded with spaces. String
length specifiers referring to output strings should count the number of bytes
available for the called function to utilize. Those that refer to input strings
can simply be the number of meaningful characters (ignoring blank padding
and null terminators), or they can count padding if the padding is actually
there.

-----GETBBV

C: void getbbv(kname, kvalue, nerr, kname_s, kvalue_s) char *kname;
char *kvalue; long int *nerr; long int kname_s; long int kvalue_s;

Note, maximum length of kvalue is 33.

FORTRAN: call getbbv(kname, kvalue, nerr) character*8 kname char-
acter*33 kvalue integer*4 nerr

note: kvalue is set in getbbv, getbbv is written in C, so the length of
kvalue is 33 instead of 32 to make room for C's null terminator.

purpose: To get a blackboard variable value from the current SAC file.

Input variables: kname: the name of the blackboard variable to read,
kname_s: (C only) string length of kname. kvalue_s: (C only) string length
of kvalue.

Output variables: kvalue: the value of the blackboard variable read, nerr:
Error flag Set to 0 if no error occurred. -----

GETFHV

C: void getfhv(kname, fvalue, nerr, kname_s) char *kname; float *fvalue;
long int *nerr; long int kname_s;

FORTRAN: call getfhv(kname, fvalue, nerr) character*8 kname real*4
fvalue integer*4 nerr

purpose: To get a floating point (real) header value from the current SAC file.

Input variables: kname: the name of the header variable to read, kname_s: (C only) string length of kname.

Output variables: fvalue: the value of the header variable read, nerr: Error flag Set to 0 if no error occurred. Set to 1336 if Header variable is undefined. Set to 1337 if Header variable does not exist. -----

GETIHV

C: void getihv(kname, kvalue, nerr, kname_s, kvalue_s) char *kname; char *kvalue; long int *nerr; long int kname_s; long int kvalue_s;

FORTRAN: call getihv(kname, kvalue, nerr) character*8 kname character*33 kvalue integer*4 nerr

note: kvalue is set in getihv, getihv is written in C, so the length of kvalue is 33 instead of 32 to make room for C's null terminator.

purpose: To get an enumerated header value from the current SAC file. Enumerated headers are character strings denoted by integer values.

Input variables: kname: the name of the header variable to read, kname_s: (C only) string length of kname kvalue_s: (C only) the maximum number of elements in the character string kvalue (must be 33 or larger, counting the null terminator).

Output variables: kvalue: Value of header field from current SAC data file. Each value represents a specific condition. To make this subroutine interface more readable, the name of the value is returned rather than its integer representation.

nerr: Error flag. Set to 0 if no error occurred. = 1336 Header variable is undefined. = 1337 Header variable does not exist. -----

GETKHV

C: void getkhv(kname, kvalue, nerr, kname_s, kvalue_s) char *kname; char *kvalue; long int *nerr; long int kname_s; long int kvalue_s;

FORTRAN: call getkhv(kname, kvalue, nerr) character*8 kname character*9 kvalue (character*17 if kname = 'kevnrm') integer*4 nerr

note: kvalue is set in getkhv, getkhv is written in C, so the length of kvalue is 9 instead of 8 (17 instead of 16) for C's null terminator.

purpose: To get a character header value from the current SAC file.

Input variables: kname: the name of the header variable to read, kname_s:
(C only) string length of kname kvalue_s: (C only) the maximum number
of elements in the character array kvalue (must be 9 or larger, counting the
null terminator (17 or longer for kevnm)).

Output variables: kvalue: Value of header field from current SAC data
file. kevnm is 17 characters long. All others are 9 (including null terminator).

nerr: Error flag. Set to 0 if no error occurred. = 1336 Header variable is
undefined. = 1337 Header variable does not exist. -----

GETLHV

C: void getlhv(kname, lvalue, nerr, kname_s) char *kname; long *lvalue;
long int *nerr; long int kname_s;

FORTTRAN: call getlhv(kname, lvalue, nerr) character*8 kname inte-
ger*4 lvalue integer*4 nerr

purpose: To get a logical header value from the current SAC file.

Input variables: kname: the name of the header variable to read, kname_s:
(C only) string length of kname

Output variables: lvalue: Value of header field from current SAC data
file.

nerr: Error flag. Set to 0 if no error occurred. = 1336 Header variable is
undefined. = 1337 Header variable does not exist. -----

GETNHV

C: void getnhv(kname, nvalue, nerr, kname_s) char *kname; long int
*nvalue; long int *nerr; long int kname_s;

FORTTRAN: call getnhv(kname, nvalue, nerr) character*8 kname inte-
ger*4 nvalue integer*4 nerr

purpose: To get an integer header value from the current SAC file.

Input variables: kname: the name of the header variable to read, kname_s:
(C only) string length of kname

Output variables: nvalue: the integer value of the header variable read.

nerr: Error flag. Set to 0 if no error occurred. = 1336 Header variable is
undefined. = 1337 Header variable does not exist. -----

NEWHDR

C: void newhdr ()

FORTTRAN: call newhdr ()

purpose: To prepare a new (default) header (does not allocate space, simply initializes everything to undefined).

SPECIAL NOTE: Except for the ones listed below, all of the header variables are set the undefined state.

NVHDR: Set to the current header version number. IFTYPE: Set to ITIME LEVEN: Set to TRUE LOVROK: Set to TRUE LCALDA Set to TRUE -----

READBBF

C: void readbbf(kname, nerr, kname_s) char *kname; long int *nerr; long int kname_s ;

FORTTRAN: call readbbf(kname, nerr) character*8 kname integer*4 nerr
purpose: To read a file of blackboard variables.

Input variables: kname: Name of disk file to read (padded at the end with blanks or '\0's). kname_s: (C only) string length of kname

Output variables: nerr: Error return flag 0 if no error occurred. [i]

RSAC1

C: void rsac1(kname, yarray, nlen, beg, del, max, nerr, kname_s) char *kname; float yarray[]; long int *nlen; float *beg; float *del; long int *max; long int *nerr; long int kname_s;

FORTTRAN: call rsac1(kname, yarray, nlen, beg, del, max, nerr) character*8 kname real*4 yarray(*) integer*4 nlen real*4 beg real*4 del integer*4 max integer*4 nerr

purpose: To read an evenly spaced SAC file.

Input variables: kname: Name of disk file to read (padded at the end with blanks or '\0's). max: number of elements in yarray. kname_s: (C only) string length of kname

Output variables: yarray: contains the data from the file. nlen: number of data points read. beg: beginning value of independent variable. del: sampling interval of the independent variable.

nerr: Error return flag 0 if no error occurred. [i] Possible values for this subroutine are: = 801 if file is not evenly spaced. = -803 if number of points in file is greater than max. In this case, the first max points are read.

RSAC2

C: void rsac2(kname, yarray, nlen, xarray, max, nerr, kname_s) char *kname; float yarray[]; long int *nlen; float xarray[]; long int *max; long int *nerr; long int kname_s;

FORTTRAN: call rsac2(kname, yarray, nlen, xarray, max, nerr) character*8 kname real*4 yarray(*) integer*4 nlen real*4 xarray(*) integer*4 max integer*4 nerr

purpose: To read an unevenly spaced or spectral SAC file.

Input variables: kname: Name of disk file to read (padded at the end with blanks or '\0's). max: number of elements in yarray and xarray. kname_s: (C only) string length of kname

Output variables: yarray: contains the dependent variable from the file. nlen: number of data points read. xarray: contains the independent variable from the file

nerr: Error return flag. Set to 0 if no error occurred. Possible values for this subroutine: = 802 If file is evenly spaced. = -803 if number of points in file is greater than max. In this case, the first max points are read.

SETBBV

C: void setbbv(kname, kvalue, nerr, kname_s, kvalue_s) char *kname; char *kvalue; long int *nerr; long int kname_s; long int kvalue_s;

FORTTRAN: call setbbv(kname, kvalue, nerr) character*8 kname character*32 kvalue integer*4 nerr

purpose: To set a blackboard variable in the current SAC file.

Input variables: kname: the name of the blackboard variable to set, kvalue: the value of the blackboard variable set, kname_s: (C only) string length of kname kvalue_s: (C only) string length of kvalue

Output variables: nerr: Error flag. Set to 0 if no error occurred.

SETFHV

C: void setfhv(kname, fvalue, nerr, kname_s) char *kname; float *fvalue;
long int *nerr; long int kname_s;

FORTRAN: call setfhv(kname, fvalue, nerr) character*8 kname real*4
fvalue integer*4 nerr

purpose: To set a floating point (real) header value in the current SAC
file.

Input variables: kname: the name of the header variable to set, fvalue:
the value of the header variable set, kname_s: (C only) string length of
kname

Output variables: nerr: Error flag. Set to 0 if no error occurred. = 1337
Header field does not exist. -----

SETIHV

C: void setihv(kname, kvalue, nerr, kname_s, kvalue_s) char *kname;
char *kvalue; long int *nerr; long int kname_s; long int kvalue_s;

FORTRAN: call setihv(kname, kvalue, nerr) character*8 kname char-
acter*32 kvalue integer*4 nerr

purpose: To set an enumerated header value in the current SAC file.
Enumerated headers are character strings denoted by integer values.

Input variables: kname: the name of the header variable to set, kvalue:
the character string value of the header variable set. kname_s: (C only)
string length of kname kvalue_s: (C only) the maximum number of elements
in the character array kvalue (must be 32 or larger, not counting the null
terminator).

Each value represents a specific condition. To make this subroutine
interface more readable, the name of the value is input rather than its integer
representation.

Output variables: nerr: Error flag. Set to 0 if no error occurred. = 1337
Header variable does not exist. -----

SETKHV

C: void setkhv(kname, kvalue, nerr, kname_s, kvalue_s) char *kname;
char *kvalue; long int *nerr; long int kname_s; long int kvalue_s;

FORTRAN: call setkhv(kname, kvalue, nerr) character*8 kname char-
acter*8 kvalue (character*16 if kname = 'kevm') integer*4 nerr

purpose: To set a character header value in the current SAC file.

Input variables: kname: the name of the header variable to set, kvalue: the character string value of the header variable set. kname_s: (C only) string length of kname kvalue_s: (C only) the maximum number of elements in the character array kvalue (must be 8 or larger, not counting the null terminator (16 or longer for kevnm)).

Note: KEVNM is 16 characters long. All others are 8.

Output variables: nerr: Error flag. Set to 0 if no error occurred. = 1337

Header field does not exist. -----

SETLHV

C: void setlhv(kname, lvalue, nerr, kname_s) char *kname; long *lvalue; long int *nerr; long int kname_s;

FORTTRAN: call setlhv(kname, lvalue, nerr) character*8 kname integer*4 lvalue integer*4 nerr

purpose: To set a logical header value in the current SAC file.

Input variables: kname: the name of the header variable to set, lvalue: New value of header field. kname_s: (C only) string length of kname

Output variables: nerr: Error flag. Set to 0 if no error occurred. = 1337

Header field does not exist. -----

SETNHV

C: void setnhv(kname, nvalue, nerr, kname_s) char *kname; long int *nvalue; long int *nerr; long int kname_s;

FORTTRAN: call setnhv(kname, nvalue, nerr) character*8 kname integer*4 nvalue integer*4 nerr

purpose: To set an integer header value in the current SAC file.

Input variables: kname: the name of the header variable to set, nvalue: the integer value of the header variable set. kname_s: (C only) string length of kname

Output variables: nerr: Error flag. Set to 0 if no error occurred. = 1337

Header field does not exist. -----

UNSETBBV

C: void unsetbbv(kname, nerr, kname_s) char *kname; long int *nerr; int kname_s;

FORTRAN: call unsetbbv(kname, nerr) character*8 kname integer*4
nerr

purpose: To unset (delete) a blackboard variable.

Input variables: kname: the name of the header variable to unset,
kname_s: (C only) string length of kname

Output variables: nerr: Error flag. Set to 0 if no error occurred. = 1337

Header field does not exist. -----

WRITEBBF

C: void writebbf(kname, nerr, kname_s) char *kname; long int *nerr;
long int kname_s ;

FORTRAN: call writebbf(kname, nerr) character*8 kname integer*4 nerr

purpose: To write a file of blackboard variables.

Input variables: kname: Name of disk file to write (padded at the end
with blanks or '\0's). kname_s: (C only) string length of kname

Output variables: nerr: error status; always 0 if there was no error.

WSAC0

C: void wsac0(kname, xarray, yarray, nerr, kname_s) char *kname; float
*xarray; float *yarray; long int *nerr; long int kname_s;

FORTRAN: call wsac0(kname, xarray, yarray, nerr) character*8 kname
real*4 xarray(*) real*4 yarray(*) integer*4 nerr

purpose: To write a SAC file to disk using current header values .

Input variables: kname: Name of disk file to write (padded at the end
with blanks or '\0's). xarray: contains the independent variable. yarray:
contains the dependent variable. kname_s: (C only) string length of kname

Output variables: nerr: error status; always 0 if there was no error.

WSAC1

C: void wsac1(kname, yarray, nlen, beg, del, nerr, kname_s) char *kname;
float yarray[]; long int *nlen; float *beg; float *del; long int *nerr; long int
kname_s;

FORTRAN: call wsac1(kname, yarray, nlen, beg, del, nerr) character*8
kname real*4 yarray(*) integer*4 nlen real*4 beg real*4 del integer*4 nerr

purpose: To write an evenly spaced SAC file.

Input variables: kname: Name of disk file to write (padded at the end with blanks or '\0's). yarray: contains the dependent variable. nlen: number of data points to write. beg: beginning value of independent variable. del: sampling interval of the independent variable. kname_s: (C only) string length of kname

Output variables: nerr: error status; always 0 if there was no error.

WSAC2

C: void wsac2(kname, yarray, nlen, xarray, nerr, kname_s) char *kname; float yarray[]; long int *nlen; float xarray[]; long int *nerr; long int kname_s;

FORTTRAN: call wsac2(kname, yarray, nlen, xarray, nerr) character*8 kname real*4 yarray(*) integer*4 nlen real*4 xarray(*) integer*4 nerr

purpose: To write an unevenly spaced or spectral SAC file.

Input variables: kname: Name of disk file to write (padded at the end with blanks or '\0's). yarray: contains the dependent variable. nlen: number of data points to write. xarray: contains the independent variable. kname_s: (C only) string length of kname

Output variables: nerr: error status; always 0 if there was no error.

Note: The I/O routines in sac.a are written in C in a FORTRAN friendly manner. This means the following details have been addressed:

- these routines can be called from either C or FORTRAN, with or without a trailing underscore (_).

- for each character string in the parameter list, there is a string length specifier in the end of the parameter list which is transparent to FORTRAN calling routines. That means that calling C routines should include the string length specifiers in the parameter list, and calling FORTRAN routines should not include them.

- the string length specifiers are declared as long int.

- the rest of the parameters are pointers in C, which means calling C routines should pass addresses, and calling FORTRAN routines should just pass the variable, because FORTRAN always passes by reference.

3 Commands

3.1 EXM: Executive Module

3.1.1 About

SUMMARY: Displays version and copywrite information.

SYNTAX: ABOUT

LATEST REVISION: January 20, 1999 (Version 0.58)

3.1.2 Cd

3.1.3 Comcor

SUMMARY: Controls SAC's command correction option.

SYNTAX: COMCOR {ON—OFF}

INPUT: ON : Turn command correction option on.

OFF : Turn command correction option off.

DEFAULT VALUES: COMCOR OFF

DESCRIPTION: SAC checks the form and content of each command you type. When it detects an error, it sends an error message to you, telling you what the error was and where it occurred. If the command correction option is on, SAC then lets you correct the command and have SAC automatically reexecute it. If this option is off, SAC merely prints the error message and returns control to you. More details and several examples are given in the User's Guide. "Command Correction Capability" in the SAC Users Manual.

LATEST REVISION: October 11, 1984 (Version 9.1)

3.1.4 Done

3.1.5 Echo

SUMMARY: Controls echoing of input and output to the terminal.

SYNTAX: ECHO ON—OFF list **where list is one or more of the following:**

ERRORS WARNINGS OUTPUT COMMANDS MACROS PROCESSED

INPUT: ON : Turn on echoing of the items in the list that follows.

OFF : Turn off echoing of the items in the list that follows.

ERRORS : Error messages generated during the execution of a command.

WARNINGS : Warning messages generated during the execution of a command.

OUTPUT : Output messages generated during the execution of a command.

COMMANDS : Raw commands as they were typed at the terminal.

MACROS : Raw commands as they appears in a macro file.

PROCESSED : Processed commands originating from the terminal or a macro file. A processed command is one where all macro arguments, blackboard variables, header variables, and inline functions have been processed (evaluated) and substituted into the raw command.

DEFAULT VALUES: ECHO ON ERRORS WARNINGS OUTPUT
OFF COMMANDS MACROS PROCESSED

DESCRIPTION: This commands lets you control which categories of the SAC input and output stream is to be echoed to the terminal or screen. There are three categories **of output:** error messages, warning messages, and output messages. There are **three categories of input:** commands typed at the terminal, commands executed from a macro, and “processed” commands. A processed command is one in which all macro arguments, blackboard variables, header variables, and inline functions have been evaluated. You can control the echoing of these categories individually. When you type a command at your terminal, the operating system normally echos each character. Thus the commands echoing option is of limited use for interactive sessions. The macro and processed options are useful when debugging a macro.

LATEST REVISION: April 21, 1989 (Version 10.4c)

3.1.6 Evaluate

SUMMARY: Evaluates simple arithmetic expressions.

SYNTAX: EVALUATE {TO TERM—name} {v} op v {op v ...} **where**
op is one of the following: ADD—SUBTRACT—MULTIPLY—DIVIDE—POWER—
SQRT—EXP—ALOG—ALOG10—SIN—ASIN—COS—ACOS—TAN—ATAN—
EQ—NE—LE—GE—LT—GT

INPUT: TO TERM : Result is written to the user's terminal.

TO name : Result is written to the blackboard variable name.

v : An floating point or integer number. Since all arithmetic is done in floating point, integers are converted to floating point numbers.

op : One of the arithmetic or logical operators listed above.

ALTERNATE FORMS:

+ may be substituted for ADD

- for SUBTRACT

* for MULTIPLY

/ for DIVIDE

** for POWER

Be sure and place spaces around each operator.

DEFAULT VALUES: EVALUATE TO TERM 1. * 1.

DESCRIPTION: This command lets you evaluate arithmetic and logical expressions. The arithmetic expression can be a compound one, containing more than one operator. In this case the expression is evaluated left to right. There is no nesting capability. A logical expression can contain only one operand. The result of evaluating this expression can be written to the user's terminal or to a specified blackboard variable. This blackboard variable can later be used directly in other commands. This is especially useful when writing macros. You can also get the value of a blackboard variable using the GETBB command.

EXAMPLES: Let's first check to see if SAC knows its multiplication tables:

u: EVALUATE 2. * 2.

s: 4.0000

u: EVALUATE 2. * 3.

s: 6.0000

Now let's convert an angle in degrees to radians and then find its tangent:

u: EVALUATE 45 * 3.14159 / 180

s: 0.78540

u: EVALUATE TAN 0.78540

s: 0.9982

That's pretty close. Notice that when you omit the first variable, SAC assumes it's value to be 1. Why does SAC not get the same answer in the following example?

u: EVALUATE TAN 45 * 3.14159 / 180

s: 0.02827

Finally let's repeat an earlier example but this time use the blackboard:

u: EVALUATE TO TEMP1 45 * 3.14159 / 180

u: EVALUATE TAN %TEMP1

s: 0.9982

LIMITATIONS: Maximum number of operators in a single expression is 10.

SEE COMMANDS: GETBB See the section on "Macros" in the SAC Users Manual.

LATEST REVISION: April 15, 1987 (Version 10.1)

3.1.7 Funcgen

SUMMARY: Generates a function and stores it in memory.

SYNTAX: FUNCGEN {type},{DELTA v},{NPTS n},{BEGIN v} **where type is one of the following:**

IMPULSE STEP BOXCAR TRIANGLE SINE {v1 v2} LINE {v1 v2}
QUADRATIC {v1 v2 v3} CUBIC {v1 v2 v3 v4} SEISMOGRAM RANDOM
{v1 v2} IMPSTRIN {n1 n2 ... nN}

INPUT: IMPULSE : Impulse at central data point.

IMPSTRIN : A series of impulses at the specified sample points.

STEP : Step function. Zero in first half. One in second half.

BOXCAR : Boxcar function. Zero in first and last thirds. One in middle third.

TRIANGLE : Triangle function. Zero in first and last quarters. Linearly increasing from zero to one in second quarter and decreasing from one to zero in third quarter.

SINE {v1 v2} : Sine wave with frequency in Hz given by v1 and phase angle in degrees given by v2. Amplitude is one. **Note that there is a factor of 2π in the phase argument:** function = $1.0 * \sin(2 * \pi * f * t)$

LINE {v1 v2} : Linear function with slope given by v1 and intercept by v2.

QUADRATIC {v1 v2 v3} : Quadratic function of the form:
,BREAK ,TEX $v1*t^2 + v2*t + v3$

CUBIC {v1 v2 v3 v4} : Cubic function of the form: ,BREAK
,TEX $v1*t^3 + v2*t^2 + v3*t + v4$

SEISMOGRAM : Sample seismogram. This is an obsolete option that has been replaced by the DATAGEN command. The DELTA, NPTS, and BEGIN options are ignored for this function. There are 1000 points in the sample seismogram.

RANDOM {v1 v2} : Random sequence (Gaussian white noise) generator. v1 is the number of random sequence files to generate and v2 is the “seed” used to generate the first random number. This seed value is stored in USER0 so that you can regenerate the same random sequence at a later time if desired.

DELTA v : Set increment between samples to v. Stored in header as DELTA.

NPTS n : Set number of data points in function to n. Stored in header as NPTS.

BEGIN v : Set begin time to v. Stored in header as BEGIN.

DEFAULT VALUES: FUNCGEN IMPULSE NPTS 100 DELTA 1.0 BEGIN 0.; Frequency and phase angle for SINE function are 0.05 and 0.

Coefficients for LINE, QUADRATIC, and CUBIC are all 1. Number of random sequences is 1 and seed is 12357.

DESCRIPTION: Executing this command is equivalent to reading a single file (except for the RANDOM option in which more than one file can be generated) into memory whose name is the name of the function generated. Any data previously in memory is destroyed. Other functions will be added as needed.

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred from the SAC data buffer to the CSS data buffer. When FUNCGEN is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

HEADER CHANGES: A header is set up in memory which accurately describes the function generated.

SEE COMMANDS: DATAGEN

LATEST REVISION: October 11, 1984 (Version 9.1)

3.1.8 Getbb

SUMMARY: Gets (prints) values of blackboard variables.

SYNTAX: GETBB {options} ALL—variable {variable ...} **where options is one or more of the following:**

TO TERMINAL—filename NAMES ON—OFF NEWLINE ON—OFF

INPUT: TO TERMINAL : Print the values to the terminal.

TO filename : Append the values to a file called filename.

NAMES [ON] : Include the name of the blackboard variable followed by an equals sign and then its value.

NAMES OFF : Only print the value of the blackboard variable.

NEWLINE [ON] : Put a newline (carriage-return) after each blackboard value printed.

NEWLINE OFF : Do not a newline after each value.

ALL : Print the values of all currently defined blackboard variables.

variable : Print the values of the specific blackboard variables listed.

DEFAULT VALUES: GETBB TO TERMINAL NAMES ON NEWLINE ON ALL

FUNCTIONAL MODULE: Executive

DESCRIPTION: The blackboard is a place to temporarily store information. This command lets you print the values of selected blackboard variables. Variables can be defined using the SETBB command. You can also use the EVALUATE command to perform basic arithmetic operations on blackboard variables and store the results in new blackboard variables. Blackboard variables can also be substituted directly into SAC commands. See the section on Macros in the Users Manual. The options to this command let you control where the values are printed and in what format to print them. You can print them to the terminal or append them to the end of a text file. You can include the variable name and an equals sign before the value or you can just have the value printed. You can have a newline placed after each value printed in a list or you can have them placed on a single line. You can use these options to make measurements on a set of data files, extract these measurements into a text file, and then read this file back into SAC using the READALPHA command to plot the results or to perform more analysis. This is illustrated in the examples section.

EXAMPLES: Assume you have already set several blackboard variables:

u: SETBB C1 2.45 C2 4.94

To later print their values you would use this command:

u: GETBB C1 C2

s: C1 = 2.45

s: C2 = 4.94

To print just their values on a single line:

u: GETBB NAMES OFF NEWLINE OFF C1 C2

s: 2.45 4.94

Assume you have a macro called GETXY that performs some type of analysis on a single data file and stores the results into two blackboard variables called X and Y. You want to perform this analysis on all of the

vertical components in the current directory, save each set of X and Y values, and plot them. In the following macro the first (and only) argument is the name of the text **file to be used to store the results:**

```
DO FILE WILD *Z
READ FILE
MACRO GETXY
GETBB TO 1 NAMES OFF NEWLINE OFF X Y
ENDDO
GETBB TO TERMINAL
READALPHA CONTENT P 1
PLOT
```

The text file would contain pairs of x-y data points, one per line, for each of the vertical data files. The final GETBB command redirecting the output back to the terminal is needed in order to close the text file and dump the buffer.

SEE COMMANDS: SETBB, EVALUATE

LATEST REVISION: Sept. 1, 1988 (Version 10.3E)

3.1.9 Help

SUMMARY: Displays information about SAC commands and features on the screen.

SYNTAX: HELP {item ...}

INPUT: item : The (full or abbreviated) name of a command, module, subprocess, feature, etc.

DEFAULT VALUES: If no item is requested, an introductory help package is displayed.

DESCRIPTION: Each requested item in the help package is displayed in the order they are requested. A short message is displayed if no information is available for an item. After a full page of output, the user is prompted to see if he or she wishes to see more information on that item. A response of “NO” or “N” will terminate the display of that item and will begin the display of the next item if any. A response of “QUIT” or “Q” will terminate

the display of all items. The help package for each command consists of the entry in the SAC Command Reference Manual. The help package for non-commands may be paragraphs from the SAC Users Manual or other information.

EXAMPLES: To get the introductory help package type:

u: HELP

Now lets say you want information on several commands:

u: HELP READ CUT BEGINDEVICE PLOT

SAC begins displaying the READ help package. After a full page, it asks if **you've seen enough:**

s: MORE?

u: YES

SAC displays the rest of the help package on READ, and then begins displaying the **help package on the CUT command:**

s: MORE?

u: NO

SAC stops displaying the CUT help package and begins displaying the BEGINDEVICE **help package:**

s: MORE?

u: QUIT

You're getting impatient so you type QUIT. SAC terminates the HELP command so you can try some of the features discussed.

ERROR MESSAGES: 1103: No help package is available.

- SAC can't find the help package. Check your SACAUX environment.

SEE COMMANDS: PRINTHELP

November 13, 1998 (Version 0.58)

3.1.10 Inicm

SUMMARY: Reinitializes all of SAC's common blocks.

SYNTAX: INICM

DESCRIPTION: This command can be used at any time to put SAC back into its initial state. All active graphics devices are terminated and the

graphics library ended. All common blocks are reinitialized to their original values. All data in memory is lost.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.1.11 Installmacro

SUMMARY: Installs macro files in the global SAC macro directory.

SYNTAX: INSTALLMACRO name {name ...}

INPUT: name : The name of a SAC macro file.

DESCRIPTION: This command lets you install your macro files into the global SAC macro directory so they can be used by anyone on your system. The location of this directory is defined by the SACAUX environmental variable, as SACAUX/macros. See the section on Macros in the SAC Users Manual.

SEE COMMANDS: MACRO

LATEST REVISION: March 20, 1992, (version 10.6e)

3.1.12 Load

SUMMARY: Load an external command.

external commands and load require extra work in the linux version of SAC.

SYNTAX: LOAD comname {ABBREV abbrevname}

INPUT: comname: The name of an external function to load from a shared object.

ABBREV abbrevname: An abbreviation or alias for comname.

DESCRIPTION: This command allows the user to load commands written to the SAC external command interface specification (See EXTERNALINTERFACE help page). This command must be a function stored in a shared object library (a .so file- see UNIX LD manpage for details). SAC will look in all shared object libraries listed in environmental variable SACSOLIST. This environmental variable should contain one or more names of shared objects in a blank delimited list. The path to these shared

objects must be specified in the LD_LIBRARY_PATH environmental variable. If SACSOLIST is not set, then SAC will look for a shared object library called libsac.so, using the paths specified in LD_LIBRARY_PATH. A library called libcom.so is distributed with SAC2000 (see EXTERNAL COMMAND ... below).

EXAMPLE:

Set up your environment to have SAC look in the current directory for a command named foo from a shared object called libbar.so. Set up an alias for foo called myfft.

```
setenv SACSOLIST "libcom.so libbar.so" setenv LD_LIBRARY_PATH
{LD_LIBRARY_PATH}: $$$add the current directory $$$to the search
path. SAC2000 SAC; load foo abbrev myfft $$$load the command SAC;
read file1.z file2.z file3.z $$$input files to pass to the command SAC; myfft
real-imag $$$invoke command with its arguments, $$$commands must parse
their own args. SAC; psp
```

How to create a shared object library containing your command(s): Solaris: cc -o libxxx.so -G extern.c foo.c bar.c

SGI cc -g -o libxxx.so -shared foo.c bar.c

LINUX: (gcc) gcc -o libxxx.so -shared extern.c foo.c bar.c sac.a

where sac.a is the sac library available where you got sac.

EXTERNAL COMMAND INCLUDED IN THE DISTRIBUTION OF SAC2000: There is one external command which is distributed with SAC. It is called FLIPXY. FLIPXY takes as input one or more X-Y datafiles, and transposes the data. This command is in libcom.so in \${SAC_AUX}/external along with the source code of FLIPXY for reference. To load FLIPXY, libcom.so must be included in SACSOLIST.

ERRORS: 1028: External command does not exist: This means that SAC did not find your external command.

This error can arise for a couple of reasons. One possibility is that your LD_LIBRARY_PATH environmental variable does not contain the path to your shared library. Another possibility is that you have not set your SACSOLIST environmental variable to contain the names of your shared libraries.

LATEST REVISION: March 21, 1996 (Version 00.50)

3.1.13 Macro

SUMMARY: Executes a SAC macro file.

SYNTAX: MACRO name {arguments}

INPUT: name : The name of the SAC macro to execute.

arguments : The arguments (if any) of the macro.

DESCRIPTION: A SAC macro is a file that contains a set of SAC commands that you want to execute as a group. You can pass arguments to the macro, define default values for these arguments, evaluate blackboard and header variables within the body of a macro, etc. The macro file can be in the current directory, in a predefined directory you specify using the SETMACRO command, or in the global SAC macro directory. See the section on Macros in the SAC Users Manual.

SEE COMMANDS: SETMACRO, INSTALLMACRO

LATEST REVISION: May 15, 1987 (Version 10.2)

3.1.14 Message

SUMMARY: Sends a message to the user's terminal.

SYNTAX: MESSAGE text

INPUT: text : Text of message to be sent. If there are blanks in the message it must be enclosed within single quotes.

DESCRIPTION: This command is useful within macro files to send status or informational messages to the user while the macro file is executing. It is not particularly useful in the interactive mode (unless you like to talk to yourself.)

EXAMPLES: To send a message without any blanks:

u: MESSAGE FINISHED

s: FINISHED

To send a message with blanks, you must use single or double quotes:

u: MESSAGE 'Job has finished.'

s: Job has finished.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.1.15 News

SUMMARY: Prints current news concerning SAC.

SYNTAX: NEWS

DESCRIPTION: If there is news to report, a “headline” will usually be written to the terminal when you start up SAC. You can then type NEWS if you want to see details.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.1.16 Pause

SUMMARY: Sends a message to the terminal and pauses.

SYNTAX: PAUSE {MESSAGE text},{PERIOD {ON—OFF—v}}

INPUT: MESSAGE text : Text of message to send to terminal before pausing. Enclose text in single quotes if it contains any blanks.

PERIOD {ON} : Turn period option on but don’t change length of pause. When this option is on, SAC pauses for a certain period of time and then resumes execution automatically.

PERIOD OFF : Turn period option off. When this option is off, SAC pauses until you type a carriage-return.

PERIOD v : Turn period option on and change length of pause to v seconds.

DEFAULT VALUES: PAUSE MESSAGE ‘Pausing’ PERIOD OFF

DESCRIPTION: This command lets you temporarily suspend the execution of a SAC macro file. When this command is executed, SAC sends a message to your terminal, pauses, and then either waits until you type a carriage return or waits for a specified period of time. This might be of interest if you wanted to study the output of a particular command before allowing the next command in the macro to be executed. It is of particular interest in the preparation of macro files to be used in demonstrations and tutorials. The ECHO command is also useful for preparing such demonstrations.

SEE COMMANDS: ECHO

LATEST REVISION: May 15, 1987 (Version 10.2)

3.1.17 Printhelp

SUMMARY: Prints hardcopies of information about SAC commands and features.

SYNTAX: PRINHELP {item ...}

INPUT: item : The (full or abbreviated) name of a command, module, subprocess, feature, etc.

DEFAULT VALUES: If no item is requested, an introductory help package is printed.

DESCRIPTION: Each requested item in the help package is printed in the order they are requested. A short message is printed if no information is available for an item. The help package for each command consists of the entry in the SAC Command Reference Manual. The help package for non-commands may be paragraphs from the SAC Users Manual or other information.

EXAMPLES: To get the introductory help package type:

u: PRINHELP

Now lets say you want information on several commands:

u: PRINHELP READ CUT BEGINDEVICE PLOT

ERROR MESSAGES: 1103: No help package is available.

- SAC can't find the help package. Check your SACAUX environment.

SEE COMMANDS: HELP

November 13, 1998 (Version 0.58)

3.1.18 Production

SUMMARY: Controls the production mode option.

SYNTAX: PRODUCTION ON—OFF

INPUT: ON : Turn production mode option on.

OFF : Turn production mode option off.

DEFAULT VALUES: PROD OFF

DESCRIPTION: When this option is on, fatal errors terminate SAC immediately. When this option is off, control is returned to the terminal after fatal errors.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.1.19 Readbbf

SUMMARY: Reads a blackboard variable file into memory.

SYNTAX: READBBF {file}

INPUT: **file** : The name of a blackboard variable file. It may be a simple filename or a relative or absolute pathname.

DEFAULT VALUES: READBBF BBF

DESCRIPTION: This command lets you read in a blackboard variable file. This file must have been previously written to disk using the WRITEBBF command. This feature lets you save information from one execution of SAC to another. You can also add coding to your own programs to access the information in these blackboard variable files. This lets you transfer information between your own programs and SAC. See the SAC Subroutines Reference Manual for details.

SEE COMMANDS: WRITEBBF, SETBB, GETBB

LATEST REVISION: May 15, 1987 (Version 10.2)

3.1.20 Report

SUMMARY: Informs the user about the current state of SAC.

SYNTAX: REPORT {list} **where list is one or more of the following:**

APF, COLOR, CUT, DEVICES, FILEID, GTEXT, HPF, LINE, MEMORY, MTW, PICKS, SYMBOL, TITLE, XLABEL, XLIM, YLABEL, YLIM

INPUT: **APF** : The name of the alphanumeric pick file.

COLOR : The current color attributes. No color table is read in until a graphics device is activated. Unless a graphics device has been activated, this report will not be correct.

CUT : The current CUT status.

DEVICES : A list of the graphics devices available on your system.

FILEID : The current file id display attributes.

GTEXT : The current graphics text attributes.

HPF : The name of the HYPO pick file.

LINE : The current linestyle attributes.

MEMORY : A dump of the available memory blocks from the memory manager. This is probably of little interest unless the memory manager is not working properly.

MTW : The current measurement time window status.

PICKS : The current time pick display attributes.

SYMBOL : The current symbol drawing attributes.

TITLE : The current plot title attributes.

XLABEL : The current x axis label attributes.

XLIM : The current x axis plot limits.

YLABEL : The current y axis label attributes.

YLIM : The current y axis plot limits.

DESCRIPTION: This command can be used to find out about the current values of certain SAC options. The values are printed to the terminal.

EXAMPLES: To get a list of the current color attributes:

u: REPORT COLOR

s: COLOR option is ON

s: DATA color is YELLOW

s: INCREMENT data color is OFF

s: SKELETON color is BLUE

s: BACKGROUND color is NORMAL

To get the names of the HYPO and card image pick files:

u: REPORT APF HPF

s: Alphanumeric pick file is MYPICKFILE

s: HYPO pick file is HYPOPICKFILE

LATEST REVISION: March 20, 1991 (Version 10.6e)

3.1.21 Setbb

SUMMARY: Sets (defines) values of blackboard variables.

SYNTAX: SETBB variable {APPEND} value {variable {APPEND} value ...}

INPUT: variable : The name of a blackboard variable. It may be a new variable or one that already has a value. The variable name can be up to 32 characters in length.

value : The new value of that blackboard variable. It must be enclosed in single or double quotes if it contains any spaces.

APPEND : Append value to the old value of variable. If this option is omitted then the new value replaces the old value.

DESCRIPTION: The blackboard is a place to temporarily store information. This information can later be accessed by the GETBB command or used directly in a command by preceeding the name of the variable with a percent sign ("%"). If you want to concatenate some other text string on the end of a blackboard variable you need to put a second percent sign at the end of the name. You can also use the EVALUATE command to perform basic arithmetic operations on blackboard variables and store the results in new blackboard variables. You can unset (delete) blackboard variables using the UNSETBB command.

EXAMPLES: To set several blackboard variables at once:

u: SETBB C1 2.45 C2 4.94

To later use these variables in a command:

u: BANDPASS CORNERS %C1 %C2

To set a blackboard variable that contains spaces:

u: SETBB MYTITLE 'Sample filter response'

To check and make sure the value is correct:

u: GETBB MYTITLE

s: MYTITLE = Sample filter response

To later use this variable in the title command it must be enclosed in quotes **and have a percent sign on both ends of the name:**

u: TITLE '%MYTITLE%'

See the section on Macros in the SAC Users Manual for more examples of the use of blackboard variables in macros.

SEE COMMANDS: GETBB, EVALUATE, UNSETBB

LATEST REVISION: May 15, 1987 (Version 10.2)

3.1.22 Setmacro

SUMMARY: Defines a set of directories to search when executing a SAC macro file.

SYNTAX: SETMACRO {MORE} directory {directory ...}

INPUT: directory : The name of a directory in which SAC macro files are stored. This may be either a relative or absolute directory name.

DESCRIPTION: This command lets you define a set of directories to search when executing SAC macro files using the MACRO command. You can define up to 100 such directories.

MORE: When the MORE option is used with setmacro, the specified directories are added to the existing list. When MORE is not used with setmacros, the existing list is replaced with the new list.

When the MACRO command is run, SAC searches for the macro in the current directory; if no file is found with the given name, SAC searches the directories listed in SETMACRO in the order that they are listed. If there are still no files found with the given name, SAC searches the global macro directory.

See the section on Macros in the SAC Users Manual.

SEE COMMANDS: MACRO

LATEST REVISION: December 5, 1996 (Version 52a)

3.1.23 Syntax

SUMMARY: Prints basic information about SAC commands.

SYNTAX: SYNTAX {item ...}

INPUT: item : The (full or abbreviated) name of a command.

DESCRIPTION: This is like the HELP command with two exceptions:

(1) Only the SUMMARY and SYNTAX sections are printed.

(2) Only commands are printed.

ERROR MESSAGES: 1103: No help package is available.

- SAC can't find the help package. Contact the programmer. **1105:**
Error reading help information for

- Usually a transient system glitch has occurred. Contact the programmer if this error persists.

SEE COMMANDS: HELP January 8, 1983 (Version 8.0)

3.1.24 Systemcommand

SUMMARY: Executes system commands from SAC.

SYNTAX: SYSTEMCOMMAND command {options}

INPUT: command : The name of the system command.

options : Any options needed by that command.

DESCRIPTION: This command allows you to execute system command while running SAC.

EXAMPLES: To get a list of files in the current UNIX directory:

u: SYSTEMCOMMAND LS

s: ... produces a list of files

LATEST REVISION: May 15, 1987 (Version 10.2)

3.1.25 Trace

SUMMARY: Controls the tracing of blackboard and header variables.

SYNTAX: TRACE [ON—OFF] name [name ...]

INPUT: ON : Turn tracing on for variables that follow.

OFF : Turn tracing off for variables that follow.

name : The name of the blackboard or header variable to trace. If this is a header variable it is of the form: filename,hdrname ,BREAK where filename is the name (or number) of the SAC data file and hdrname is the name of a SAC header variable.

DEFAULT VALUES: TRACE ON

DESCRIPTION: This command can be used to trace or track the values of SAC blackboard or header variables while SAC is executing. It is useful primarily for debugging long or complicated macros. When the tracing for a variable is turned on, its current value is printed. While the tracing is on, its value is checked after the execution of each command. Each time its value changes a new output line is printed. When tracing is turned off, its current value is also printed.

EXAMPLES: To turn tracing on for the blackboard variable called TEMP1 and for the **header variable called T0 belonging to the file called MYFILE:**

```
u: TRACE ON TEMP1 MYFILE,T0
s: TRACE (on) TEMP1 = 1.45623
s: TRACE (on) MYFILE,T0 = UNDEFINED
```

As you execute commands, either typed at the terminal or executed from a macro, SAC will check the values of the variables versus the saved value and print a message whenever either one of them changes. Assume that some calculations are performed that caused TEMP1 to change and T0 to become **defined**. **SAC would print the messages:**

```
s: TRACE (mod) TEMP1 = 2.34293
s: TRACE (mod) MYFILE,T0 = 10.3451
```

Later in the processing TEMP1 may change again:

```
s: TRACE (mod) TEMP1 = 1.93242
```

When the tracing is turned off, SAC will print the current value one last **time:**

```
u: TRACE OFF TEMP1 MYFILE,T0
s: TRACE (off) TEMP1 = 1.93242
s: TRACE (off) MYFILE,T0 = 10.3451
```

LATEST REVISION: January 27, 1989 (Version 10.4B)

3.1.26 Transcript

SUMMARY: Controls output to the transcript files.

SYNTAX: TRANSCRIPT options **where options are one or more of the following:**

OPEN—CREATE—CLOSE—CHANGE—WRITE FILE filename CONTENTS ALL—list MESSAGE text **where list is one or more of the following:**

ERRORS WARNINGS OUTPUT COMMANDS MACROS PROCESSED

INPUT: OPEN : Open and append transcript to the bottom of an existing file.

CREATE : Create a new transcript file.

CLOSE : Close an open transcript file.

CHANGE : Change the contents of an open transcript file.

WRITE : Write message to transcript file without changing its status or contents.

FILE filename : Define the name of a transcript file.

MESSAGE text : Write message contained in text to transcript file. This message can be used to identify the processing being done or to identify different events as they are being processed. This message is NOT retained between executions of this command.

CONTENTS ALL : Define the contents of the transcript file to be all input/output types.

CONTENTS list : Define the contents of the transcript file. This is a list of the types of input and output to include in the file.

ERRORS : Error messages generated during the execution of a command.

WARNINGS : Warning messages generated during the execution of a command.

OUTPUT : Output messages generated during the execution of a command.

COMMANDS : Raw commands as they were typed at the terminal.

MACROS : Raw commands as they appears in a macro file.

PROCESSED : Processed commands originating from the terminal or a macro file. A processed command is one where all macro arguments, black-board variables, header variables, and inline functions have been processed

(evaluated) and substituted into the raw command.

DEFAULT VALUES: TRANSCRIPT OPEN FILE TRANSCRIPT CONTENTS ALL

DESCRIPTION: A transcript file can be used to record the results of executing SAC. It can be a complete or partial transcript. It can contain the results from one or more executions. You can have up to five transcripts active at any given time, each keeping track of different aspects of the execution. One use as illustrated below is to record the commands typed at the terminal and to later use this as a macro file.

EXAMPLES: To create a new transcript file called MYTRAN containing everything except **the processed commands:**

u: TRANSCRIPT CREATE FILE MYTRAN CONTENTS ERRORS WARNINGS OUTPUT COMMANDS MACROS

If later during this session you did not want the macro commands to be sent **to this file you would use the CHANGE option:**

u: TRANSCRIPT CHANGE FILE MYTRAN CONTENTS ERRORS WARNINGS OUTPUT COMMANDS

To define a transcript file called MYRECORD which records the commands as **they are typed at the terminal:**

u: TRANSCRIPT CREATE FILE MYRECORD CONTENTS COMMANDS

Later this file, perhaps after some editing, could be used as a macro to automatically execute the same set of commands. In the final example assume you needed to process a number of events overnight. You could set up transcript files for each of these events (with different names) that recorded the results of the processing. In addition you could store any error messages from the processing of all of these events in a single transcript **file:**

u: TRANSCRIPT OPEN FILE ERRORTRAN CONTENTS ERRORS

u: TRANSCRIPT WRITE MESSAGE 'Processing event 1'

These commands would be placed in the macro that processes each event. It is assumed that the name of the event is passed into the macro as the first argument. By using the open option, the message and any errors would be appended to the end of the file. By examining this error transcript the next

day, you could quickly see whether any errors occurred during processing and for which events these errors occurred.

LATEST REVISION: January 27, 1989 (Version 10.4B)

3.1.27 Unsetbb

SUMMARY: Unsets (deletes) blackboard variables.

SYNTAX: UNSETBB ALL — variable ...

INPUT: ALL : Unset all of the currently defined blackboard variables.

variable : Unset the blackboard variable variable.

DESCRIPTION: The blackboard is a place to temporarily store information. Blackboard variables are defined using the SETBB and EVALUATE commands. They can be accessed by the GETBB command or used directly in a command by preceeding the name of the variable with a percent sign ("%"). This command allows you to unset previously defined blackboard variables. You may unset all variables or only a named subset.

EXAMPLES: To unset several blackboard variables at once:

u: UNSETBB C1 C2 X

To unset all blackboard variables:

u: UNSETBB ALL

SEE COMMANDS: SETBB, EVALUATE, GETBB

LATEST REVISION: January 26, 1989 (Version 10.4B)

3.1.28 Writebbf

SUMMARY: Writes a blackboard variable file to disk.

SYNTAX: WRITEBBF {file}

INPUT: file : The name of a blackboard variable file. It may be a simple filename or a relative or absolute pathname.

DEFAULT VALUES: WRITEBBF BBF

DESCRIPTION: This command lets you write a blackboard variable file to disk. It can later be read into SAC using the READBBF command. This feature lets you save information from one execution of SAC to another. You can also add coding to your own programs to access the information in

these blackboard variable files. This lets you transfer information between your own programs and SAC. See the SAC Subroutines Reference Manual for details.

SEE COMMANDS: READBBF, SETBB, GETBB

LATEST REVISION: May 15, 1987 (Version 10.2)

3.2 DFM: Data File Module

3.2.1 Chnhdr

SUMMARY: Changes the values of selected header fields.

SYNTAX: CHNHDR { file n1 n2 ... } field v {field v ... }

INPUT: file : This is an optional keyword that can be followed by a list of numbers indicating which file's headers are to be changed.

n1 n2 ...: Integers indicating which file's headers to change.

field : The name of a SAC header variable. These variables are listed in an appendix to this manual. Also, field may be the keyword ALLT as discussed below. **Note:** in order to maintain internal consistency, the following header variables cannot be changed with CHNHDR: NVHDR, NPTS, NWFID, NORID, and NEVID.

v : Set the value of that field to v. The type of the field and its new value must match. Use single quotes for alphanumeric fields with embedded blanks. Use TRUE or FALSE for logical fields. YES or NO are also acceptable for logical fields. Use variable names (see appendix) for value fields. For offset time fields (B, E, O, A, F, and Tn), v may also be of the form: GMT v1 v2 v3 v4 v5 v6 where v1, v2, v3, v4, v5, and v6 are the GMT year, julian day, hour, minute, second, and millisecond of the time. If v1 is a two digit number, SAC will assume it is in the current century, unless that would mean that the year is in the future yet, in which case, SAC assumes the previous century. To be certain you get what you want, use four digits.

UNDEF : Use this keyword instead of v to "undefine" a header field.

ALLT v : Add v seconds to all defined header times. Subtract v seconds from the zero time.

DESCRIPTION: This command lets you change any of SAC's header fields. A specific file or list of files can be changed by specifying the integer value(s) corresponding to the order in which the file(s) were read in. If no integer filelist is specified, all files in memory will have their header fields changed. To change the headers of the files on disk follow this command with the WRITE or WRITEHDR command. SAC does some validity checking on the new values but you may want to verify the results using the LISTHDR command. There is a set of six variables in the header (NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, and NZMSEC) which contain the reference or "zero" time of the file. This is the only GMT in the SAC header. All other times in the header (B, E, O, A, F, and Tn) are offsets in seconds relative to this reference time. You may change the reference time and all of the defined offset times by using the "ALLT v" option. That number of seconds are added to each defined offset time. That same number of seconds is also subtracted from the reference time. This preserves the actual GMT time of the data. As a convenience, you may enter a GMT time instead of a relative time when changing the offset times. When the GMT time is entered it is converted to a relative time before storing it in the offset time field. A description of each of the SAC header fields is contained in part 6 of the user manual. To see this in sac type "help file.format".

EXAMPLES: To define the event latitude, longitude and name in all the files in memory:

u: CHNHDR EVLA 34.3 EVLO -118.5

u: CHNHDR KEVNM 'LA goes under'

To define the event latitude, longitude and name in files 2 and 4:

u: CHNHDR file 2 4 EVLA 34.3 EVLO -118.5

u: CHNHDR file 2 4 KEVNM 'LA goes under'

To change the event type to earthquake:

u: CHNHDR IEVTYP IQUAKE

To set the first arrival time to its undefined state:

u: CHNHDR A UNDEF

Assume you know the GMT origin time of an event and that you want

to quickly change all the times in the header so that this origin time is the zero or reference time and all other offset times are correct relative to this time. **First set the origin time using the GMT option:**

u: CHNHDR O GMT 1982 123 13 37 10 103

Now use the LISTHDR command to find out what O is relative to the current **reference time:**

u: LISTHDR O

s: O 123.103

Now use the ALLT option to subtract this value from all of the offset times and add it to the reference time. You also want to change the field that describes the type of reference time stored in these files.

u: CHNHDR ALLT -123.103 IZTYPE IO

Notice the minus sign because you are subtracting this value from the offset times.

HEADER CHANGES: Potentially all header fields.

ERROR MESSAGES: 1006: Length of string variable exceeded at symbol

- Alphanumeric header field too long. **1301:** No data files read in.

SEE COMMANDS: LISTHDR, WRITE, WRITEHDR

LATEST REVISION: January 8, 1983 (Version 8.0)

3.2.2 Convert

SUMMARY: Converts data files from one format to another.

SYNTAX: CONVERT {FROM} {format} infile {TO {format} outfile}—{OVER {format}}

where format is one of the following: SAC—ALPHA

INPUT: infile : The name of the input data file.

outfile : The name of the output data file.

OVER : Overwrite the input data file.

SAC : SAC formatted binary data file.

ALPHA : Alphanumeric equivalent of SAC binary data file.

DEFAULT VALUES: CONVERT FROM SAC infile OVER SAC

DESCRIPTION: This command converts a single data file from one format to another. It will be replaced in a future update with an improved system that will use the READ and WRITE commands.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.2.3 Copyhdr

SUMMARY: Copies header variables from one file in memory to all others.

SYNTAX: COPYHDR {FROM name—n} hdrlist

INPUT: FROM name : Copy header list from named file in memory.

FROM n : Copy header list from numbered file in memory.

hdrlist : Space delimited list of header variables to copy.

DEFAULT VALUES: COPYHDR FROM 1

DESCRIPTION: This command lets you copy the values of any SAC header variable from one file in memory to all of the remaining files in memory. You can select which file you want to copy from.

EXAMPLES: Assume you are using PPK to mark several times in the header of a file called FILE1. You are using the header variables T3 and T4. To copy those same **markers into files FILE2 and FILE3:**

u: READ FILE1

u: PPK

u: ... use cursor to mark times T3 and T4.

u: READ MORE FILE2 FILE3

u: COPYHDR FROM 1 T3 T4

In this next example, assume you have read in a large number of files and you want to copy the event location, EVLA and EVLO, from the file called ABC into all of the other headers. This can be easily done by referencing the file by **name not number:**

u: COPYHDR FROM ABC STLA STLO

HEADER CHANGES: Potentially all.

LATEST REVISION: May 15, 1987 (Version 10.2)

3.2.4 Cut

SUMMARY: Defines how much of a data file is to be read.

SYNTAX: CUT {ON—OFF—pdw—SIGNAL}

INPUT: ON : Turn cut option on but don't change pdw.

OFF : Turn cut option off.

pdw : Turn cut option on and change pdw. A pdw is a partial data window. It consists of a starting and a stopping value of the independent variable, usually time, which defines which part of a **file you wish to read. The most general form of this pdw is:** ref offset ref offset

ref : A reference value, one of the following: Z—B—E—O—A—F—N—T_n, where n=0,1...9. The meanings of these names are given below.

offset : A positive or negative number which is added to the reference value.

SIGNAL : Equivalent to typing: A -1 F +1.

DEFAULT VALUES: Cut option is off. If the start or stop offset is omitted it is assumed to be zero. If the start reference value is omitted it is assumed to be Z. If the stop reference value is omitted it is assumed to be the same as the start reference value.

DESCRIPTION: With the cut option off, the entire file is read. With it on, only that portion of the file between the starting and stopping cut values is read. These are values in terms of the independent variable in the data file, normally time. See the section on "Data File Structure" in the Users Manual for a discussion of dependent and independent variables. The following **mnemonics are used to represent certain values of the independent variable:**

B: Disk file beginning value.

E: Disk file ending value.

Z: Zero.

N: Interpret offset in terms of number of data points not independent variable. Only used for stop values.

The following mnemonics refer to reference values stored in the SAC header and are used primarily to cut time files:

O: Event origin time.

A: First arrival time.

F: Signal ending time.

Tn: User defined time picks, $n = 0,1...9$

The above mnemonics plus an optional positive or negative offset define either a starting or stopping value. O, A, F, and Tn can be defined for a given data file using the CHNHDR command. A and F can also be defined using the automatic picker (APK) or manual pick plot (PLOT PK) commands. If you want to select the same time window from a group of data files that have different reference times, you must use the SYNCHRONIZE command before executing the CUT and READ commands. SYNCHRONIZE modifies the headers so that each file has the same reference time. It also adjusts all of the relative times, including B and E. Then when the files are cut, they will have the same time window. Since CUT is applied to the headers on disk, you must use the WRITEHDR command after the SYNCHRONIZE command and before the READ command to get the correct results. You can also pad the beginning or end of a file with zeros by turning on the FILLZ option in the CUTERR command, defining a cut that extends beyond the current limits of the file, and then reading the file into memory using the READ command.

EXAMPLES: In these examples we assume time is the independent variable and seconds are the units.

B E: Disk begin to disk end—same as turning cut off.

B 0 30: First 30 secs of the disk file.

A -10 30: From 10 secs before to 30 secs after first arrival.

B N 2048: First 2048 points of disk file.

30.2 48: From 30.2 to 48 secs relative to disk file zero.

ERROR MESSAGES: 1322: Undefined starting cut for file

- undefined reference value in the header record. - this error can be controlled by use of CUTERR command. - when this error is off, the disk begin value is used. **1323:** Undefined stop cut for file

- undefined reference value in the header record. - this error can be controlled by use of CUTERR command. - when this error is off, the disk

end value is used. **1324:** Start cut less than file begin for file

- bad CUT parameters. - this error can be controlled by use of CUTERR command. - when this error is off, the disk begin value is used or zeros are inserted at the beginning of the data. **1325:** Stop cut greater than file end for file

- bad CUT parameters. - this error can be controlled by use of CUTERR command. - when this error is off, the disk end value is used or zeros are inserted at the end of the data. **1326:** Start cut greater than file end for file

- bad CUT parameters. - this error cannot be turned off.

SPECIAL NOTE: Since this is a parameter-setting command, the above errors will not appear until the READ command is executed. Also, some of the above errors can be converted to warnings by the use of the CUTERR command.

LIMITATIONS: There is currently no provision for cutting unevenly-spaced files or spectral files.

SEE COMMANDS: READ, APK, PLOTPK, SYNCHRONIZE, CUTERR

LATEST REVISION: January 8, 1983 (Version 8.0)

3.2.5 Cuterr

SUMMARY: Controls errors due to bad cut parameters.

SYNTAX: CUTERR FATAL—USEBE—FILLZ

INPUT: FATAL : Treat cut errors as fatal.

USEBE : Replace bad start cut with file begin and bad stop cut with file end.

FILLZ : Fill with zeros before file begin or after file end to account for difference between bad cut and file begin and end.

DEFAULT VALUES: FILLZ for signal stacking subprocess, USEBE for others.

DESCRIPTION: This command controls error conditions arising from bad cut parameters. These error conditions can be defined as fatal errors (FATAL). If the starting or ending cut parameter is undefined in the header record the disk file beginning or ending value can be chosen (USEBE). If

the cut parameter is defined but the resulting cut value is less than the disk beginning value or greater than the disk ending value, the beginning or ending values can be used, or enough zeros can be added before or after the actual data to make up the difference (FILLZ).

EXAMPLES: Assume that FILE1 has a begin time, B, of 25 seconds, a first arrival, A, of 40 seconds, and a sampling rate of 100 samples per second. Assume the **following commands were typed:**

u: CUT A -20 E

u: READ FILE1

The starting cut evaluates to 20 seconds and generates a bad cut error condition. In the USEBE mode, the starting value would become 25 seconds (B). In the FILLZ mode, 500 zeros (5 seconds at 100 samples per second) would be inserted before the data and the starting value would remain as 20 seconds.

SEE COMMANDS: CUT, READ

LATEST REVISION: January 8, 1983 (Version 8.0)

3.2.6 Cutim

SUMMARY: Cuts files in memory. Can cut multiple segments from each file.

SYNTAX: CUTIM pdw [pwd ...]

INPUT: **pdw** : Partial Data Window. It consists of a starting and a stopping value of the independent variable, usually time, which defines which part of a file you wish to read. The most general form of this pdw
is: ref offset ref offset

ref : A reference value, one of the following: Z—B—E—O—A—F—T_n, where n=0,1...9. The meanings of these names are given below.

offset : A positive or negative number which is added to the reference value.

DEFAULT VALUES: If the start or stop offset is omitted it is assumed to be zero. If the start reference value is omitted it is assumed to be Z. If the stop reference value is omitted it is assumed to be the same as the start

reference value.

DESCRIPTION: Where the CUT command simply sets cut points and waits for the next READ, CUTIM actually carries out the cut at the time the command is given. The user can READ a file, and type CUTIM with the desired cutpoints, and SAC will cut the file to those specified cutpoints without further disc-access. CUTIM also allow multiple pairs of cutpoints. The user can READ three files into SAC, and use CUTIM with four sets of cutpoints; the result will be 12 files in memory. The start and stop values are given in terms of the independent variable in the data file, normally time. See the section on “Data File Structure” in the Users Manual for a discussion of dependent and independent variables. The following mnemonics are used to represent certain values of the independent variable:

B: Disk file beginning value.

E: Disk file ending value.

Z: Zero.

Note: the N option is not available for CUTIM.

The following mnemonics refer to reference values stored in the SAC header and are used primarily to cut time files:

O: Event origin time.

A: First arrival time.

F: Signal ending time.

Tn: User defined time picks, $n = 0, 1 \dots 9$

The above mnemonics plus an optional positive or negative offset define either a starting or stopping value. O, A, F, and Tn can be defined for a given data file using the CHNHDR command. A and F can also be defined using the automatic picker (APK) or manual pick plot (PLOT PK) commands. If you want to select the same time window from a group of data files that have different reference times, you must use the SYNCHRONIZE command before executing the CUTIM command. SYNCHRONIZE modifies the headers so that each file has the same reference time. It also adjusts all of the relative times, including B and E. Then when the files are cut, they will have the same time window.

EXAMPLES: In these examples we assume time is the independent variable and seconds are the units.

B E: begin to end—same as not cutting.

B 0 30: First 30 secs of the file.

A -10 30: From 10 secs before to 30 secs after first arrival.

30.2 48: From 30.2 to 48 secs relative to disk file zero.

If a stop time is on a pick, and the next start time is an offset from zero,
avoid ambiguity by giving the stop time an offset of zero:

T1 T2 0 1000 1500

otherwise, the 1000 will be taken as the offset of the T2, and SAC will complain that the second start time (1500) is missing a stop time.

ERROR MESSAGES: 1322: Undefined starting cut for file

- undefined reference value in the header record. - this error can be controlled by use of CUTERR command. - when this error is off, the disk begin value is used. **1323:** Undefined stop cut for file

- undefined reference value in the header record. - this error can be controlled by use of CUTERR command. - when this error is off, the disk end value is used. **1324:** Start cut less than file begin for file

- bad CUT parameters. - this error can be controlled by use of CUTERR command. - when this error is off, the disk begin value is used or zeros are inserted at the beginning of the data. **1325:** Stop cut greater than file end for file

- bad CUT parameters. - this error can be controlled by use of CUTERR command. - when this error is off, the disk end value is used or zeros are inserted at the end of the data. **1326:** Start cut greater than file end for file

- bad CUT parameters. - this error cannot be turned off.

SPECIAL NOTE: Also, some of the above errors can be converted to warnings by the use of the CUTERR command.

LIMITATIONS: There is currently no provision for cutting unevenly-spaced files or spectral files.

SEE COMMANDS: CUT, READ, APK, PLOTPK, SYNCHRONIZE, CUTERR

LATEST REVISION: June 18, 1999 (Version 0.58)

3.2.7 Datagen

SUMMARY: Generates sample data files and stores them in memory.

SYNTAX: DATAGEN {MORE} {COMMIT—ROLLBACK—RECALLTRACE}
{SUB name} {filelist}

where name is one of the following:

LOCAL REGIONAL TELESEISEM

INPUT: MORE : Place the new sample data files in memory AFTER the old data. If this option is omitted, the new sample data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to generating more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before generating more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

SUB name : Select the sub-directory name from which to read the data. Where the subdirectory name is local, regional, or teleseismic.

A filelist is required. Possible filenames are listed below.

name : LOCAL—REGIONAL—TELESEIS ,BREAK Specifics about the contents of these sub-directories is given below.

filelist : A list of SAC sample data files. The names of the files in each subdirectory are given below. This list may contain simple filenames and wildcard characters. See the READ and WILD commands for a complete

description.

NOTE: The files provided were generated on the Sun which is a bigendian machine. On a littleendian machine (Alpha or Linux), these files will be unreadable. In the aux directory, there is a SAC macro called tosac, which reproduces these files from the ascii files which are also provided.

DEFAULT VALUES: DATAGEN COMMIT SUB LOCAL cdv.z

DESCRIPTION: This command reads one or more files from a set of sample seismograms provided with the program. In fact, it operates much like the READ command except that it gets the data from a special data directory. You specify the name of the sub-directory containing the event of interest. Each sub-directory contains a different type of event. You can use wildcards just like the READ command to specify groups of files in the sub-directory.

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred from the SAC data buffer to the CSS data buffer. When DATAGEN is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

ERROR MESSAGES: 1301: No data files read in.

- haven't given a list of files to read. - none of the files in the list could be read. **1314:** Data file list can't begin with a number.

1315: Maximum number of files in data file list is

WARNING MESSAGES: 0101: opening file

0108: File does not exist:

0114: reading file

- Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the READERR command.

LOCAL EVENT: The local event occurred in the Livermore Valley of California. It was a small unfelt event (ML 1.6). It was recorded by the Livermore Local Seismic Network (LLSN). LLSN is a set of vertical and three-component stations operated by LLNL and the USGS. Data from nine

three-component stations are included in this set. There is 40 seconds of data sampled at 100 samples per second. Station information, event information, p-wave time picks, **and coda picks are included in the headers. The filenames are:**

cal.z, cal.n, cal.e
cao.z, cao.n, cao.e
cda.z, cda.n, cda.e
cdv.z, cdv.n, cdv.e
cmn.z, cmn.n, cmn.e
cps.z, cps.n, cps.e
cva.z, cva.n, cva.e
cvl.z, cvl.n, cvl.e
cvy.z, cvy.n, cvy.e

REGIONAL EVENT: The regional event occurred in Nevada and was recorded by the Digital Seismic Network (DSS). DSS is a set of four broadband **three-component stations in the Western U.S. The stations are:**

elk: Elko, NV
lac: Landers, CA
knb: Kanab, UT
mnv: Mina, NV

The sampling rate is 40 samples per second. The files contain 300 seconds of data, starting 5 seconds before the origin time of the event. The filenames **are:**

elk.z, elk.n, elk.e
lac.z, lac.n, lac.e
knb.z, knb.n, knb.e
mnv.z, mnv.n, mnv.e

TELESEISMIC EVENT The teleseismic event occurred off the coast of Northern California near Eureka on September 10, 1984. It was a moderate to large event (ML 6.6, MB 6.1, MS 6.7) and was felt from the San Francisco Bay area to Roseburg, Oregon. It was recorded at the Regional Seismic Test

Network (**RSTN**), a set of five stations in the U.S. and Canada. The stations are:

cpk: Tennessee

ntk: Northwest Territories, Canada

nyk: New York

onk: Ontario, Canada

sdk: South Dakota

Both mid-period and long period data is included. Data from cpk was not available and the long-period data from sdk is clipped. There is 1600 seconds of data in this set. The long-period data was recorded at 1 sample per second **and the mid-period data at 4 samples per second. The filenames are:**

ntkl.z, ntkl.n, ntkl.e, ntkm.z, ntkm.n, ntkm.e

nykl.z, nykl.n, nykl.e, nykm.z, nykm.n, nykm.e

onkl.z, onkl.n, onkl.e, onkm.z, onkm.n, onkm.e

sdkl.z, sdkl.n, sdkl.e, sdkm.z, sdkm.n, sdkm.e

SEE COMMANDS: READ, WILD, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.2.8 Deletechannel

SUMMARY: Deletes one or more files from the file list.

SYNTAX: [D]ELETE[C]HANNEL ALL [D]ELETE[C]HANNEL filename—filenumber—range {filename—filenumber—range ... }

INPUT: ALL: Deletes all files from memory. The user need not specify filenames or filenumbers

filename: Name of a file in the file list.

filenumber: Number of a specific file in the file list. The first file in the list is 1, the second is 2, etc. (The command FILENUMBER ON tells SAC to display the file numbers in most of the plots.)

range: Two file numbers separated by a dash: eg. 11-20.

TYPE: Action-taking

EXAMPLES: dc 3 5 * deletes 3rd and 5th file. dc SO01.sz SO02.sz * deletes named files. dc 11-20 * deletes all the files from * the 11th through the 20th, * inclusive. dc 3 5 11-20 SO01.sz SO02.sz * deletes all of the above.

ERROR MESSAGES: 5106: File name not in file list **5107:** File number not in file list

SEE COMMANDS: DELETSTACK, FILENUMBER

3.2.9 Headerwindow

3.2.10 Listhdr

SUMMARY: Lists the values of selected header fields.

SYNTAX: LISTHDR {listops} {hdrlist} **where listops are one or more of the following:**

DEFAULT—PICKS—SPECIAL FILES ALL—list COLUMNS 1—2 INCLUSIVE ON—OFF

INPUT: DEFAULT : Use the default list, which includes all defined header fields.

PICKS : Use the picks list, which includes those header fields used to define time picks.

SPECIAL : Use the special user defined list.

FILES ALL : List headers from all files in data file list.

FILES list : List headers from a subset of the files in the data file list. The subset is defined as a list of file numbers.

COLUMNS 1 : Format output into a single column of entries.

COLUMNS 2 : Format output into two columns.

INCLUSIVE : ON includes header variables which are undefined. OFF excludes them.

hdrlist : List of header fields to be included in the special list.

DEFAULT VALUES: LISTHDR DEFAULT FILES ALL COLUMNS 1 INCLUSIVE OFF

DESCRIPTION: The user can define which items to list or can use either of two standard lists. The first list (DEFAULT) contains all of the

header fields. The second list (PICKS) contains those header fields which are directly or indirectly used to define time picks. This list contains the **following fields:** B, E, O, A, Tn, KZTIME, KZDATE. More standard lists can be added if needed. A special list can be defined by the user at any time and can then be requested again by using the SPECIAL option. The full listing for a header field consists of its name, an equals sign, and its current value. Some of the fields for some files will be undefined. SAC stores a special value in those fields to flag them as undefined. The listing excludes these undefined fields unless the INCLUSIVE option is ON. For integers and floats the undefined value is -12345; for character strings and those integers which are used to indicate character strings, the undefined value is "UNDEFINED".

A description of each of the SAC header fields is contained in an appendix to this manual.

ERROR MESSAGES: 1301: No data files read in.

EXAMPLES: To get a two column listing of the time picks only:

u: LISTHDR PICKS COLUMNS 2

To get a default listing of the third and fourth files in the data file list:

u: LISTHDR FILES 3 4

To list the values of the begin and end time only:

u: LISTHDR B E

To define a special list of the station parameters:

u: LISTHDR KSTNM STLA STLO STEL STDP

To reuse this special list later during the same execution:

u: LISTHDR SPECIAL

LATEST REVISION: December 12, 1996 (Version 52a)

3.2.11 Pickauthor

SUMMARY: Tell sac to read author list (and possibly phase pick information) from a user-defined preferences file, or interactively enter author

list on the PICKAUTHOR command line.

SYNTAX: PICKAUTHOR author1 {author2 author3 ... } PICKAUTHOR FILE {filename} PICKAUTHOR PHASE {filename}

INPUT: authorlist: sac uses the input to create the author list.

FILE: if the FILE option is used, sac will read the author list from the preferences file. If a filename is given on the command line, sac will read the specified file, else it will read the most recently entered file name from a previous call to PICKAUTHOR. If no filename was ever entered, sac will look for SACAUX/csspickprefs.

PHASE: this option behaves essentially like the FILE option with the added benefit of having sac read specific header variable information as well.

DEFAULT VALUES: PICKAUTHOR FILE

DESCRIPTION: PICKAUTHOR is provided as a means to override the preferences file on the command line. It can be used to provide a prioritized list of authors at the command line, or to redirect SAC from one preferences file to another. For more on the preferences files, see PICKPREFS and READCSS.

Note: If the user alters the preference settings while data is in the data buffers, the picks in the SAC data buffer (the buffer visible to the user through LISTHDR and CHNHDR etc.) may be modified. Eg. if the author list is “john rachel michael” and some files are read with the READCSS command some arrivals may be read with author = michael. (The user will probably not be aware of who the author is for a given pick, because the author field in CSS does not appear in the SAC format.) If the user later uses the PICKAUTHOR command to change the author list to “peter doug rachel”, then on a READCSS MORE command, no arrivals with author = michael will be read from the data files, and the file already in memory will lose the picks which have michael as the author. The user may not know why seemingly random picks have disappeared. For an explanation, type HELP PICKPREFS.

SEE COMMANDS: PICKPREFS, READCSS, PICKPHASE

3.2.12 Pickphase

SUMMARY: Tell sac to read phase pick information (and possibly the author list) from a user-defined preferences file, or interactively enter phase pick information on the PICKPHASE command line.

SYNTAX: PICKPHASE header phase author {header phase author ...}
} PICKPHASE FILE {filename} PICKPHASE AUTHOR {filename}

INPUT: **header:** name of a header variable: t0 - t9. **phase:** name of phase of pick desired for the given header variable. **author:** name of the author desired for the given header or hyphen, “-”, to tell sac to use the author list.

FILE: if the FILE option is used, sac will read the phase pick info. from the preferences file. If a filename is given on the command line, sac will read the specified file, else it will read the most recently entered file name from a previous call to PICKPHASE. If no filename was ever entered, sac will look for SACAUX/csspickprefs.

PHASE: this option behaves essentially like the FILE option with the added benefit of having sac read the author list as well.

DEFAULT VALUES: PICKPHASE FILE

DESCRIPTION: PICKPHASE is provided as a means to override the preferences file on the command line. It can be used to provide specific header/phase/author information at the command line, or to redirect SAC from one preferences file to another. For more on the preferences files, see PICKPREFS and READCSS.

Note: If the user alters the preference settings while data is in the data buffers, the picks in the SAC data buffer (the buffer visible to the user through LISTHDR and CHNHDR etc.) may be modified. Eg. if the allowed phases include pP and PKiKP when some SAC files are read with the READ command which have some pP picks or some PKiKP picks these picks would be present in the Tn markers. If PICKPHASE is later used to remove pP and PKiKP from the allowed phases before the next READCSS MORE call, then pP and PKiKP arrivals will not be read from the CSS files, and the pP and PKiKP picks in the existing data will be removed from

the Tn markers. For an explanation, type `HELP PICKPREFS`.

SEE COMMANDS: `PICKPREFS`, `READCSS`, `PICKAUTHOR`

3.2.13 Pickprefs

SUMMARY: The `PICKPREFS` command is used to control the way that SAC manages and or loads picks from a variety of input data formats (e.g., `CSS`, `GSE`, `SUDS` etc...) into the time marker variables `T0` to `T9` (aka. `Tn`). When this option is `OFF` (the default), the picks loaded into the time markers correspond to the first picks that SAC finds in the input data. If this options is `ON`, SAC will use the preferences file described in the `READCSS` command.

Note: Because of the structured nature of the preferences file (which aligns specific phases with specific marker variables), and the free flowing nature of the interactions without the preferences, a change in the `PICKPREFS` in the middle of processing can change the picks in the datafiles. See the description below for details.

SYNTAX: `PICKPREFS ON PICKPREFS OFF PICKPREFS`

INPUT: ON: instructs SAC to pass arrivals from the `CSS` buffer through the preferences file on its way to the SAC buffer. This is useful in macros that require specific arrivals to be in specific `Tn` header variables.

OFF: instructs SAC to bypass the preferences file and load the first 10 picks it encounters for a given file. This is the default. It allows the user to be aware of picks s/he may not be aware of with the `PICKPREFS ON`.

If now option is provided on the commandline, `PICKPREFS` will toggle the use of preferences file `ON` or `OFF`.

DEFAULT VALUES: `PICKPREFS OFF`

DESCRIPTION: Since version 0.58, `sac2000` has had two different header buffers: one formatted according to the SAC file format, and one formatted according to the relational `CSS` 3.0 file format. Adding the `CSS` data buffer has made it easier to read relational formats such as `CSS`, `GSE`, and `SUDS`. Having two buffers has allowed **the process management commands:** `COMMIT`, `ROLLBACK`, and `RECALLTRACE`.

One drawback of having these two buffers is the complexity of moving arrivals from the dynamic CSS arrival table to the rather ridged T0 - T9 picks in the SAC format. This problem was solved in version 0.58 by setting in place a preferences file called `csspickprefs`. This file is in the `aux` directory and can be overridden by writing one of your own. For more information about how to use the `csspickprefs` file, use `HELP READCSS`. For details on how to override the default preferences file, use `HELP PICKAUTHOR` or `HELP PICKPHASE`.

The drawback of using the preferences file was that it would only accept phase names and/or author names listed in the preferences file or those entered at the command line with `PICKPHASE` or `PICKAUTHOR`. In other words, if a CSS data file from either a flat-file, or the Oracle database has a pP arrival, and pP is not specified in the preferences file, the user would never know that the pP is there. The pP pick will be read into the CSS data buffer in SAC, but it will not be transferred to the SAC data buffer, and will not participate in any of the SAC commands. It may be written out by the `WRITECSS` command, or it may get flushed out during a `COMMIT` command, and be lost entirely.

The solution we have worked out is to allow the user to bypass the preferences file. In version 0.59, the default is to read the first 10 available picks from the CSS buffer directly into the SAC buffer whenever data is transferred from the one to the other. By use of this new command, `PICK-PREFS`, the user can tell SAC to use the preferences file. This is useful if the user has a macro which expects to find a specific phase in a specific Tn header variable.

Data is transferred from the CSS buffer to the SAC buffer on any `READCSS`, `READDB` (Oracle version of SAC only), `READGSE`, or `READSUDS` command, as well as `COMMIT`, `ROLLBACK`, and `RECALLTRACE`. `COMMIT`, `ROLLBACK`, or `RECALLTRACE` get called by **default by any of the following commands:** - any command that writes data (`WRITE`, `WRITECSS`, `WRITEGSE`, etc.) - any command that reads data with the `MORE` option specified - the `SORT` command.

If the user alters `PICKPREFS` and or the preference settings while data

is in the data buffers, the picks in the SAC buffer may be modified. Eg. if PICKPREFS is OFF (the default) when some SAC files are read with the READ command they may have some pP picks or some PKiKP picks which would be present in the Tn markers. If PICKPREFS is later turned OFF, for a READCSS (or READDB) call, if pP and/or PKiKP aren't listed in the preferences file, then pP and PKiKP arrivals will not be read from the CSS files, and the pP and PKiKP picks in the existing data will be removed from the Tn markers.

SEE COMMANDS: READCSS, READDB, PICKAUTHOR, PICK-PHASE, COMMIT, ROLLBACK, RECALLTRACE

3.2.14 Read

SUMMARY: Reads data from SAC data files on disk into memory.

SYNTAX: READ [options] [filelist] **where options is one or more of the following:**

MORE TRUST ON—OFF COMMIT—ROLLBACK—RECALLTRACE
DIR CURRENT—name XDR ALPHA SEG Y SCALE [ON—OFF]

ALL options MUST precede any element in the filelist.

INPUT: MORE : Place the new data files in memory AFTER the old ones. If this option is omitted, the new data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

TRUST ON—OFF : This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous

versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the **RECALLTRACE option:** - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of header variables which are committed, and which are rolled back.)

DIR CURRENT : Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name : Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

XDR : The input files are in XDR format. This format is used for moving binary data files to/from a different architecture, such as a pc running LINUX.

ALPHA : The input files are SAC formatted alphanumeric (ascii) files. the ALPHA option is incompatible with the XDR option.

SEGY : Read file formatted according to the IRIS/PASSCAL form of the SEG Y format. This format allows one waveform per file.

SCALE : Used only in conjunction with the SEG Y option, the SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the SEG Y files. When SCALE is ON, SAC multiplies counts by a SCALE factor given in the file. This scale option changes with the SCALE options of READCSS, READGSE, READDB, and READSUDS. If SCALE is OFF, the SCALE value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The preferred method is with the TRANSFER command. It is recommended to leave SCALE off for

the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

filelist : file—wild ...

file : A legal filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one. See the DESCRIPTION and EXAMPLES sections below for more details.

wild : A wildcard laden token that expands to a list of filenames. See the DESCRIPTION and EXAMPLES sections below and the WILD command for more details.

DEFAULT VALUES: READ COMMIT DIR CURRENT

DESCRIPTION: All commands in SAC work on the data that is currently in memory. This data in memory is analogous to the temporary or working files used by a text editor. The READ command transfers data from one or more disk files into memory. The default is to read all of the data from each disk file. The CUT command can be used to specify that only a portion of each disk file be read. SAC files produced in or after the year 2000 are presumed to have a four digit value for the year. Files with two digit year values will be assumed to be in the twentieth century, and will be incremented by 1900. Normally all data in memory prior to the execution of another READ command is lost. The new data replaces the old data. If the keyword MORE is the second symbol in the command, the new data is placed in memory after the old data. The data file list becomes the concatenation of the old file list and the new file list. **There are three cases where the MORE option may be useful:**

- (1) The filelist is too long to be typed on one line.
- (2) A name was misspelled in a long filelist.
- (3) A file is read, some analysis performed, and a comparison with the original is desired.

Examples of each of these cases are given below. The filenames may be simple filenames in the current directory or they may be absolute or relative pathnames pointing to other directories on your **system**. **Examples of absolute pathnames are:**

UNIX: /dir/subdir/file

VMS: disk:[dir.subdir]file

PRIMOS: ;disk;dir;subdir;file

Examples of relative pathnames are:

UNIX: subdir/file

PRIMOS: *;subdir;file

In the above examples “disk” is the name of a physical disk partition, “dir” is the name of a top level directory, “subdir” is a subdirectory of that partition, and “file” is a file in that subdirectory. In general there is no limit on the nesting of subdirectories. Filenames may also contain wild-card characters. You can use them match a single character, to match zero or more characters, and to form groupings of characters. Some examples are given below. See the WILD command for more examples and a complete explanation of all the wildcarding options.

*** Important *** SAC has two data buffers; this is what allows SAC to provide the COMMIT, ROLLBACK and RECALLTRACE commands. One data buffer stores the header information in SAC format, and the second stores headers in CSS 3.0 format. This CSS 3.0 data buffer allows seamless consistency with CSS 3.0 in READCSS and WRITECSS; it also allows direct access to the CSS 3.0 formatted Oracle database. In CSS (a relational format), it is important to maintain consistency with the event IDs (evid, or nevid in SAC). In SAC format (a very flat format), such consistency is not as important, and in some cases, it is lost. Anytime data is loaded into SAC, it is stored in both buffers. When transferring data from SAC to CSS data buffers, there is a potential ambiguity in handling event information. If matching evids are found, it could be that the two files have identical event information, or it could be that the match is an artifact of the merge of these two different data formats within SAC. Two peices of information are involved in resolving this ambiguity, one is the history of data loaded into SAC memory, and the other is the confidence the user sets with the TRUST ON—OFF option on the command line of most Read commands and ADDSTACK. It is expected that the user will have some idea if the data files are consistent, if they share event information, etc. The history of data

loaded into SAC memory begins when data is loaded into memory without the MORE option, and ends the next time data is loaded into memory without the MORE option. Any time in between that data is loaded into memory with the MORE option, it becomes part of the existing history. All commands which load data into memory are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. The READDB command is the most reliable way to load data into SAC because the database insures consistency. For this reason, the levels of confidence (in ascending order) are LOW, HIGH, and RDB. TRUST is an option set on the commandline of most commands that load SAC data. TRUST can be ON or OFF. Each time a command loads data into SAC, it responds to both the confidence level and the TRUST to determine how to treat matching event IDs. When requisite levels of TRUST and confidence are present, matching event IDs are treated as an indicator that the two files share identical event information. This being the case, the event IDs are left unchanged. When requisite levels of TRUST and confidence are not met, matching event IDs are **are treated as artifacts of the merging of two different data formats:** SAC and CSS. READDB, being the only truly reliable way to load data, will always treat the data as reliable as long as READDB is the only method used to load data into SAC. In this case, the TRUST will not be used, and the confidence lever will always be set to RDB. If any other data loading method is used, then the confidence will be reduced to HIGH, or LOW, and RDB will respond similarly to the next set of commands. **The following commands are considered HIGH confidence:** READ, READCSS, READHDR, and ADDSTACK. These commands will consider both the confidence level and the TRUST in determining how to handle event ID matches. If the confidence is HIGH (or RDB) and the TRUST is ON, then confidence is set to HIGH, and the event IDs are treated as reliable. However if the TRUST is OFF or if the confidence level is LOW, then the confidence is set to LOW and the matching event IDs are treated as artifacts, and new IDs are generated for the incoming data file. The following commands are considered LOW confidence because event ID **information is not available:** READTABLE, READGSE, READSUDS,

FUNCGEN, DATAGEN, READSP, READSDD, and READ with the SEGY option and READCSS when the input files are in the CSS 2.8 data format. These commands will always generate event IDs and set the confidence level to LOW. Commands written by the user and added to SAC via the external command interface (the LOAD command) are a special case, since the user writes the code. In these cases, the confidence is based entirely on the TRUST that the user sets. Hence, it is incumbent upon the user to set the TRUST either on the commandline or within the code. For more information use HELP EXTERNAL. Within a given history, if data is loaded into SAC by any means other than READDDB, the data is probably not consistent with the database, and should probably not be loaded back into the database, unless the user takes responsibility to insure consistency among events.

EXAMPLES: In the following examples it is assumed that the following SAC data **files are in your current disk directory:** F01, F02, F03, and G03. In these examples, the UNIX wildcard characters (e.g., “?” matches any single character and “*” matches zero or more characters) are used. See the WILD command for more information on how to use wildcards. To read the first three **files:**

u: READ F01 F02 F03

The following command produces the same result using the wildcard operator:

u: READ F*

This command also produces the same result by using the concatenation **operator:**

u: READ F0[1,2,3]

To read the second, third, and fourth files:

u: R F02 ?03

The following examples show the use of the MORE option:

u: R F03 G03

... files F03 and G03 are in memory.

u: R F01 F02

... files F01 and F02 are in memory.

u: R MORE F03 G03

... files F01, F02, F03, and G03 are in memory.

This example uses the MORE option when a filename was misspelled:

u: R F01 G02 F03

s: **WARNING: File does not exist: G02**

s: Will read the remainder of the data files.

... files F01 and F03 are in memory.

u: R MORE F02

... files F01, F03, and F02 are now in memory.

... note the order of the files in this case.

If you wanted to apply a highpass filter to a data file and then graphically **compare the results to the original:**

u: READ F01

u: HIGHPASS CORNER 1.3 NPOLES 6

u: READ MORE F01

u: PLOT1

... plot shows filtered and original data

Now assume you were in the directory “/me/data” when you started up SAC and that you wanted to work with the data files in the subdirectories “event1” and “event2”:

u: READ DIR EVENT1 F01 F02

... files in directory /me/data/event1 are read.

u: READ F03 G03

... files in same directory are read.

u: READ DIR EVENT2 *

... all files in /me/data/event2 are read.

u: READ DIR CURRENT F03 G03

... files in directory /me/data are read.

Note: For examples of the differing behavior between the COMMIT, ROLLBACK, RECALLTRACE options, see the commands of the same names.

ERROR MESSAGES: 1301: No data files read in.

- haven't given a list of files to read. - none of the files in the list could be read. **1320:** Available memory too small to read file

1314: Data file list can't begin with a number.

1315: Maximum number of files in data file list is

6002: No more data-sets available.

WARNING MESSAGES: 0101: opening file

0108: File does not exist:

0114: reading file

- Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the **READERR** command.

HEADER CHANGES: E, DEPMIN, DEPMAX, DEPMEN, B if cut option is on.

SEE COMMANDS: CUT, READERR, WILD, COMMIT, ROLL-BACK, RECALLTRACE

LATEST REVISION: June. 18, 1999 (Version 0.58)

3.2.15 Readalpha

3.2.16 Readcss

SUMMARY: Read data files in CSS external format from disk into memory.

NOTE: The READCSS command reads flat files which adhere to CSS 3.0 or **2.8 data formats. The following tables are supported for version 3.0:**

wfdisc, wftag, origin, arrival, assoc, sitechan, site, affiliation, origerr, origin, event, sensor, instrument, gregion, stassoc, remark sacdata.

For version 2.8 only wfdisc, arrival, and origin are supported. Previous versions of READCSS required that the origin file have only one line which would be associated with the waveforms pointed to by the wfdisc file. The current version can extract the correct origin (or origins) using information from a wftag file or using an evid column in the wfdisc file (position 284

- 291). If no such information is available, READCSS will default to its previous behavior, and use the first row in the origin file. There is now no information lost when data is read using READCSS. Although existing SAC commands can only access a subset of the CSS data, everything read from CSS flatfiles is retained in memory and will be written to disk with the WRITECSS command.

READCSS now reads a non-standard table named `sacdata` (written by the WRITECSS command) which holds data from the SAC header that does not have a place in the standard schema. With the `sacdata` table, there is now no information loss when SAC data is written in CSS format and then re-read. For instance, you can now write frequency domain data to disk with WRITECSS and re-read it later with READCSS.

READCSS now has a binary option that allows it to read binary CSS files written by WRITECSS. In binary mode the `css` options have no effect. That is, the entire contents of the specified file(s) will be read.

READCSS supports the following binary datatypes: On bigendian machines (eg. Sun) `t8`, `t4`, `f8`, `f4`, `s4`, `s3`, `s2`, `i4`, `i2`, `g2`, `e1`, and `ri` (real-imag).

On littleendian machines (eg. DEC or PC) `f8`, `f4`, `t8`, `t4`, `i4`, `i2`, `s4`, `s2`, and `g2`

SYNTAX: READCSS {BINARY—ASCII} {MAXMEM `v`} {MORE} {TRUST ON—OFF} {VERBOSE ON—OFF} {SHIFT ON—OFF} {SCALE ON—OFF} {COMMIT—ROLLBACK—RECALLTRACE} {DIR `name`} `wfdisc` `list` {`filelist`} {`css options`}

The `css options` are one or more of the following:

STATION `station` CHANNEL `channel` BANDWIDTH `band` code ORIENTATION `orientation code`

which causes this command to further select from files that are qualifying members of `filelist` based on the content of their corresponding records in the `wfdisc` file.

INPUT: ASCII: (Default) Reads normal ASCII flatfiles.

BINARY: Reads binary CSS files. See the WRITECSS command for more information on this format.

TRUST ON—OFF: This option is used to resolve an ambiguity in

converting files from SAC to CSS format. When converting the data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

MAXMEM: Specify the maximum fraction of physical memory to use when reading large data sets. When this limit is reached, no more waveforms will be read, although other tables may still be read. The default value for MAXMEM is 0.3.

MORE : See the READ command.

VERBOSE ON—OFF: If VERBOSE is ON, SAC displays extended information about the waveforms being read, and prints a summary of the CSS tables that were filled. SAC also displays a progress indicator for the conversion to SAC internal format.

SHIFT ON—OFF: If SHIFT is on, the origin time is set to zero, and other time related header variables are shifted back to be consistent with the origin time. Some of the distance related header variables are also affected. SHIFT ON is the default.

SCALE ON—OFF: The SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the .w file. When SCALE is ON, SAC multiplies counts by a SCALE factor given as CALIB in the .wfdisc file. This scale option changes with the SCALE options of READ SEG, READGSE, READDB, and READSUDS. If SCALE is OFF, the CALIB value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The preferred method is with the TRANSFER command. It is recommended to leave SCALE off for the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous

versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

DIR name : The directory to be searched for wfdisc(s).

wfdisclist : The name(s) of one or more wfdisc files.

filelist : A list of data file names contained in the previously specified wfdisc(s). These files will be searched for first in the directory specified with the DIR option, then using the path specified in the wfdisc. If no filelist is supplied, all the data files defined in the specified wfdisc(s) will be read into memory.

STATION station : station is a string of six or fewer characters. Selects those lines from the .wfdisc file whose KSTNM matches station. station may contain * and ? wildcards.

CHANNEL channel : channel is a string of eight or fewer characters. Selects those lines from the .wfdisc file whose channel specifier matches channel. channel may contain * and ? wildcards.

BANDWIDTH type : A 1-letter code. Usual values are E Extremely Short Period S Short Period H High Broad Band B Broad Band M Mid Period L Long Period V Very Long Period U Ultra Long Period R Extremely Long Period

Selects those files whose ‘chan’ field has a leading character which is s, m or l. The character * selects all.

ORIENTATION type : Usual values are: Z N E (Vertical North

East) A B C (Triaxial along edges of cube standing on corner) 1 2 3 Orthogonal but non-standard orientation

Selects those files whose ‘chan’ field has a final character which matches code. The character * selects all.

MAGNITUDE MB—MS—ML—DEF: Determines which value of magnitude to put into SAC2000’s magnitude header variable. MB gets the bodywave magnitude, MS gets the surfacewave magnitude, ML gets the local magnitude, **and def (the default) follows this algorithm:** if Ms exists and is greater than or equal to 6.6, take it, else, if Mb exists take it, else, if Ms exists take it, else take ML.

DEFAULT VALUES: READCSS * ASCII MAXMEM 0.3 VERBOSE
OFF COMMIT STATION * BAND * CHAN * ORIENT *

DESCRIPTION: See the READ command. Oct. 27, 1998 (Version 00.58)

All commands which load data into memory have are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. For READCSS, when the confidence is HIGH that all the data files are consistent in the numbering of event IDs, matching event IDs are treated as having identical event information. When the confidence is LOW in READCSS, matching event IDs are understood as an artifact, and new event IDs are generated for the incoming file. For more details use HELP READ.

How READCSS reads picks from the .arrival file:

SAC2000 has two data buffers. One holds the data in SAC format, and one holds it in CSS3.0 format. READCSS reads all the available arrivals into the CSS buffer. Only 10 picks will fit into the SAC formatted buffer. The command PICKPREFS controls the way the picks are transferred from the CSS buffer to the SAC buffer.

There is a preferences file which SAC2000 uses to determine which phases and authors’ picks to transfer between buffers. The default preferences file is \$SACAUX/csspickprefs. This default can be overridden by either the PICKAUTHOR or PICKPHASE commands. These commands can select a user-defined preferences file, or they can interactively override the prefer-

ences file.

If PICKPREFS is OFF (the default), SAC will transfer the first 10 picks from the CSS data buffer to the SAC data buffer. If PICKPREFS is ON, SAC will transfer the picks according to the preferences file, or the PICKAUTHOR and PICKPHASE commands.

The following is an example of a preferences file:

```
john rachel michael
```

```
t0 P - t1 Pn rachel t2 Pg - t3 S - t4 Sn - t5 Sg - t6 Lg - t7 LR - t8 Rg -  
t9 pP -
```

Note: phase names are case sensitive; author names are not.

The first few lines are a prioritized list of author names (analysts who have made picks) that sac can use to select picks from the data. The remaining lines tell sac which css phase picks should be mapped into which sac header variables (T0 through T9). A hyphen (-) in the third column tells sac to use the prioritized author list. An optional author name can be specified in the third column which will override the default author list for this pick.

For a given waveform, sac will choose from the available picks those which match the given phase and author name. If an author name is specified in the third column, sac will try to match that; if it fails to match that author name, it will leave the header variable undefined. If the third column has a hyphen, sac will try to match the first name in the list; if it fails it will try to match the second name and so on until it gets a match, or the author list is exhausted (in which case the header variable is left undefined). In the example file shown above, T0 will have a P phase with john, rachel, or micheel as the author, or it will be left blank; T1 will have a Pn phase and rachel as the author, or it will be left blank. For each pick header variable there is a corresponding string header variable (KT0 through KT9). These will be populated with the phase names of the corresponding picks.

The basic format of the preferences file is: Author names are delimited by newlines. There are no blank lines before the first author name, and no blank spaces at the begining of a line. There are no blank spaces in the middle of an author name. Author names must be unique. Author

names may be up to 15 characters long. There may be any number of author names. The names are listed in order of priority, the most important author name first. The last name in the author list is followed by an empty line to designate the end of the author list.

The header variable information occupies 10 lines in three columns. The first column simply lists the names of the header variables in numerical order. The second column lists specific phase names; phase names can be up to eight characters long. The third column can have a specific author name, or a hyphen. The columns are separated by tabs. There are no spaces anywhere in these 10 lines.

SEE COMMANDS: READ, PICKPREFS, PICKAUTHOR, PICK-PHASE, COMMIT, ROLLBACK, RECALLTRACE

3.2.17 Readerr

SUMMARY: Controls errors that occur during the READ command.

SYNTAX: READERR {BADFILE FATAL—WARNING—IGNORE},
{NOFILES FATAL—WARNING—IGNORE} {MEMORY SAVE—DELETE}

INPUT: BADFILE : Errors that occur when the file could not be read or didn't exist.

NOFILES : None of the files in the read filelist could be read.

FATAL : Make error condition fatal. Send error message and stop processing the command.

WARNING : Send warning message but continue processing the command.

IGNORE : Ignore condition and continue processing the command.

MEMORY : Action on files in memory if no files could be read.

DELETE : This MEMORY option indicates that files previously in memory are to be deleted.

SAVE : This MEMORY option indicates that files previously in memory are to remain in memory.

DEFAULT VALUES: READERR BADFILE WARNING NOFILES
FATAL MEMORY DELETE

DESCRIPTION: Several errors can occur when you try to read a data file into memory using the READ command. The file may not exist or it may exist but can't be read. When SAC encounters one of these bad files, it normally sends a warning message and then tries to read the rest of the files in the filelist. If you want SAC to stop reading in files whenever a bad file is encountered set the BADFILE condition to FATAL. If you don't even want to see the warning message, set the BADFILE condition to IGNORE. If none of the files in the filelist could be read, SAC normally sends an error message and stops processing. If you want SAC to send a warning message or ignore this problem completely, set the NOFILES condition accordingly. Also, any files previously in memory can be deleted (removed from) or remain in memory by using the MEMORY DELETE or MEMORY SAVE options. The CUTERR command can be used to control certain errors that occur due to bad cut parameters.

SEE COMMANDS: READ, CUTERR

LATEST REVISION: March 20, 1992 (Version 10.6e)

3.2.18 Readgse

SUMMARY: Read data files in GSE 2.0 format from disk into memory.

Note: GSE data enters SAC via SAC's CSS data buffers. To understand how arrivals are handled, use HELP READCSS and HELP PICKPREFS.

SYNTAX: READGSE {MAXMEM v} {MORE} {VERBOSE ON—OFF} {SHIFT ON—OFF} {SCALE ON—OFF} {COMMIT—ROLLBACK—RECALLTRACE} {DIR name} filelist

INPUT: MAXMEM: Specify the maximum fraction of physical memory to use when reading large data sets. When this limit is reached, no more waveforms will be read, although other tables may still be read. The default value for MAXMEM is 0.3.

MORE : See the READ command.

VERBOSE ON—OFF: If VERBOSE is ON, SAC displays extended information about the waveforms being read, and prints a summary of the CSS tables that were filled. SAC also displays a progress indicator for the

conversion to SAC internal format.

Note: the SHIFT option is moot at this point. For the time being, origin information is not read because it cannot be associated with a waveform. The release of GSE 2.1 format should allow us to make the association, then we will be able to read origin, and the SHIFT option will have meaning. */ **SHIFT ON—OFF:** If SHIFT is on, the origin time is set to zero, and other time related header variables are shifted back to be consistent with the origin time. Some of the distance related header variables are also affected. SHIFT ON is the default.

SCALE ON—OFF: The SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the GSE file. When SCALE is ON, SAC multiplies counts by a SCALE factor given as CALIB in the GSE file. This scale option changes with the SCALE options of READ SEG Y, READCSS, READDB, and READSUDS. If SCALE is OFF, the CALIB value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The preferred method is with the TRANSFER command. It is recommended to leave SCALE off for the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of

which header variables are committed, and which are rolled back.)

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

DIR name : The directory to be searched for gsefile(s).

filelist : The name(s) of one or more gse files.

DEFAULT VALUES: READGSE * MAXMEM 0.3 VERBOSE OFF
COMMIT

DESCRIPTION: See the READ command. October 27, 1998 (Version 00.58)

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred from the SAC data buffer to the CSS data buffer. When READGSE is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

NOTES:

The following GSE Data messages can be read: WAVEFORM
STATION CHANNEL ARRIVAL

Waveform formats of INT, CM6, and CM8 can be read.

Arrivals, although read, will not appear in SAC since the DETECTIONS message is not yet read, and without a DETECTION ID, arrivals cannot be associated with channels.

LATEST REVISION: April 22, 1999 (Version 00.58)

3.2.19 Readhdr

SUMMARY: Reads headers from SAC data files into memory.

SYNTAX: READHDR [options] [filelist] **where options is one or more of the following:**

MORE TRUST ON—OFF COMMIT—ROLLBACK—RECALLTRACE
DIR CURRENT—name

ALL options MUST precede any element in the filelist.

INPUT: MORE : Place the new data file headers in memory AFTER

the old ones. If this option is omitted, the new data file headers REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

TRUST ON—OFF : This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting the data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR CURRENT : Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name : Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

filelist : file—wild ...

file : A legal filename. This may be a simple filename or a pathname.

The pathname can be a relative or absolute one. See the DESCRIPTION and EXAMPLES sections of the READ command for more details.

wild : A wildcard laden token that expands to a list of filenames. See the DESCRIPTION and EXAMPLES sections of the READ command and the WILD command for more details.

DESCRIPTION: This command reads the headers from a set of SAC files into memory. You can then list the header contents (LISTHDR), change header values (CHNHDR), and then write the headers back to disk (WRITEHDR). This is much faster than reading entire files into memory, when only the headers are needed.

All commands which load data into memory have are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. For READHDR, when the confidence is HIGH that all the data files are consistent in the numbering of event IDs, matching event IDs are treated as having identical event information. When the confidence is LOW in READHDR, matching event IDs are understood as an artifact, and new event IDs are generated for the incoming file. For more details use HELP READ.

ERROR MESSAGES: 1301: No data files read in.

- haven't given a list of files to read. - none of the files in the list could be read. **1314:** Data file list can't begin with a number.

1315: Maximum number of files in data file list is

1335: Illegal operation—only data file headers in memory.

- only LISTHDR, CHNHDR, and WRITEHDR operations. can be performed after a READHDR.

WARNING MESSAGES: 0101: opening file

0108: File does not exist:

0114: reading file

- Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the READERR command.

SEE COMMANDS: READ, LISTHDR, CHNHDR, WRITEHDR, READERR, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.2.20 Readsdd

SUMMARY: Reads data from SDD data files on disk into memory.

SYNTAX: READSDD [options] [filelist] **where options is one or more of the following:**

MORE COMMIT—ROLLBACK—RECALLTRACE DIR CURRENT—name
ALL options MUST precede any element in the filelist.

INPUT: MORE : Place the new data files in memory AFTER the old ones. If this option is omitted, the new data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR CURRENT : Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name : Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

filelist : file—wild ...

file : A legal filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one.

wild : A wildcard laden token that expands to a list of filenames. more details.

DEFAULT VALUES: READ COMMIT DIR CURRENT

TYPE: Action-producing

FUNCTIONAL MODULE: Data File

DESCRIPTION: All the same restrictions apply to READSDD as to the READ command. See the READ command DESCRIPTION and EXAMPLES sections for more detail.

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred from the SAC data buffer to the CSS data buffer. When READSDD is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.2.21 Readsuds

SUMMARY: Read data files in PC-SUDS format from disk into memory.

Note: SUDS data enters SAC via SAC's CSS data buffers. To understand how arrivals are handled, use HELP READCSS and HELP PICK-PREFS.

SYNTAX: READSUDS {MAXMEM v} {MORE} {VERBOSE ON—OFF} {SHIFT ON—OFF} {COMMIT—ROLLBACK—RECALLTRACE} {DIR name} filelist

INPUT: MAXMEM: Specify the maximum fraction of physical memory to use when reading large data sets. When this limit is reached, no more waveforms will be read, although other tables may still be read. The default value for MAXMEM is 0.3.

MORE : See the READ command.

VERBOSE ON—OFF: If VERBOSE is ON, SAC displays extended information about the waveforms being read, and prints a summary of the CSS tables that were filled. SAC also displays a progress indicator for the conversion to SAC internal format.

SHIFT ON—OFF: If SHIFT is on, the origin time is set to zero, and other time related header variables are shifted back to be consistent with the origin time. Some of the distance related header variables are also affected. SHIFT ON is the default.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

DIR name : The directory to be searched for sudsfile(s).

filelist : The name(s) of one or more suds files.

DEFAULT VALUES: READSUDS * MAXMEM 0.3 VERBOSE OFF COMMIT

DESCRIPTION: See the READ command. Oct. 27, 1998 (Version 00.58)

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred from the SAC data buffer to the CSS data buffer. When READSUDS is used, the con-

fidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

NOTES:

READSUDS assumes that the data are still in PC byte-order, and swaps bytes as necessary while reading the files.

The following SUDS headers should be populated: DESCRIP-TRACE STATIONCOMP FEATURE EVENT ORIGIN

Statident structs for a given channel must have all fields set identically **to allow joining:** i.e. `dt->dt_name = fe->fe_name = sc->sc_name`.

There should be only 1 origin and 1 event in the SUDS file since PC-SUDS has no way to associate origins with features or descriptraces.

`ev->number` must equal `or->number` to associate the event with the origin.

SUDS Magnitude, Authority, Program, Instrument, and Phase codes must be from the following code lists in order to translate to CSS.

Suds magnitude codes: =====
or->mag_type in origin (type char): case 'S': "ms" case 'b': "mb"
 case 'c': "md" case 'l': "ml" case 'm': "mw" case 's': "ms" case 'w':
 "mw" =====

Suds Authority codes: =====
or->authority in origin (type short): `ev->authority` in event
 case 1000: "USGS-Menlo-Park" case 1002: "CALNET" case
 1050: "RTP-USGS-Menlo-Park" case 2000: "Geophysical-Institute-U-
 of-Alaska" case 3000: "University-of-Washington" case 4000: "Lamont-
 Doherty-Geological-Observatory" case 5000: "IRIS" case 5100: "GSN"
 case 5200: "ASRO" case 5300: "PASSCAL" case 6000: "LLNL" case
 7000: "LBL" case 8000: "LANL" =====

Suds program codes =====
or->program in origin (type char):
 case '7': "Hypo-71" case 'h': "HypoInverse" case 'l': "HypoLayer"
 case 'c': "Centroid" case 'v': "Velest" =====

Suds event codes =====

ev-¿ev_type in event (type char):

case ‘e’: “ke” /*known earthquake */ **case ‘E’:** “qb” /*quarry
blast */ **case ‘n’:** “kn” /*known nuclear explosion */ **case ‘i’:** “iq”
/*icequake*/ **case ‘r’:** “rq” /*regional earthquake*/ **case ‘t’:** “tq”
/*teleseismic earthquake*/ **case ‘K’:** “kr” /*known rockburst*/ **case**
‘k’: “sr” /*suspected rockburst*/ **case ‘m’:** “sm” /*suspected mine ex-
plosion*/ **case ‘M’:** “km” /*known mine explosion*/ **case ‘s’:** “se”
/*suspected earthquake*/ **case ‘S’:** “sn” /*suspected nuclear explosion*/
case ‘l’: “ls” /*landslide*/ **case ‘d’:** “si” /*suspected induced event*/
case ‘D’: “ki” /*known induced event*/ **case ‘x’:** “sx” /*suspected ex-
perimental explosion*/ **case ‘X’:** “kx” /*known experimental explosion*/

Suds instrument codes =====

suds_statident-¿inst_type (type short):

case 0: “Unk” **case 1:** “sp-usgs” **case 2:** “sp-wwssn” **case 3:**
“lp-wwssn” **case 4:** “sp-dwwssn” **case 5:** “lp-dwwssn” **case 6:** “hglp-
lamont” **case 7:** “lp-hglp-lamont” **case 8:** “sp-sro” **case 9:** “lp-sro” **case**
10: “sp-asro” **case 11:** “lp-asro” **case 12:** “sp-rstn” **case 13:** “lp-rstn”
case 14: “sp-uofa-U-of-Alaska” **case 15:** “STS-1/UVBB” **case 16:** “STS-
1/VBB” **case 17:** “STS-2” **case 18:** “FBA-23” **case 19:** “Wilcoxin”
case 50: “USGS-cassette” **case 51:** “GEOS” **case 52:** “EDA” **case**
53: “Sprengnether-refraction” **case 54:** “Teledyne-refraction” **case 55:**
“Kinematics-refraction” **case 300:** “amplifier” **case 301:** “amp/vco”
case 302: “filter” **case 303:** “summing-amp” **case 304:** “transmit-
ter” **case 305:** “receiver” **case 306:** “antenna” **case 307:** “battery”
case 308: “solar-cell” **case 309:** “discriminator” **case 310:** “discr-
rack” **case 311:** “paper-recorder” **case 312:** “film recorder” **case**
313: “smoked glass recorder” **case 314:** “atod convertor” **case 315:**
“computer” **case 316:** “clock” **case 317:** “time receiver” **case 318:**
“magnetic tape” **case 319:** “magntic disk” **case 320:** “optical disk”

suds phases in fe-¿feature (type short) =====

case 0: “none” **case 1:** “window” **case 2:** “f finis” **case 3:** “Max-

Amp” case 50: “P-first” case 51: “P” case 52: “P*” case 53: “PP”
case 54: “PPP” case 55: “PPPP” case 56: “PPS” case 57: “Pg”
case 58: “Pn” case 59: “Pdiff” case 60: “PcP” case 61: “PcPPKP”
case 62: “PcS” case 63: “pP” case 64: “pPP” case 65: “PKP” case
66: “PKPPKP” case 67: “PKPPKS” case 68: “PKPSKS” case 69:
“PKS” case 70: “pPKS” case 71: “PKKP” case 72: “PKKS” case
73: “PcPPKP” case 74: “PcSPKP” case 100: “S-first” case 101: “S”
case 102: “S*” case 103: “SS” case 104: “SSS” case 105: “SSSS”
case 106: “Sg” case 107: “Sn” case 108: “ScS” case 109: “SPcS”
case 110: “sS” case 111: “sSS” case 112: “sSSS” case 113: “SScS”
case 114: “ScSPKP” case 115: “ScP” case 116: “SKS” case 117:
“SKKS” case 118: “SKKKS” case 119: “SKSSKS” case 120: “SKP”
case 121: “SKKP” case 122: “SKKKP” case 201: “Lg” case 202:
“Lr” case 203: “Lr2” case 204: “Lr3” case 205: “Lr4” case 206:
“Lq” case 207: “Lq2” case 208: “Lq3” case 209: “Lq4” case 301: “t”

=====

SEE COMMANDS: READ, PICKAUTHOR, PICKPHASE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: October 27, 1998 (Version 00.58)

3.2.22 Sort

SUMMARY: Sorts files in memory by header fields.

SYNTAX: SORT COMMIT—ROLLBACK—RECALLTRACE header {ASCEND—DESCEND} {header {ASCEND—DESCEND} ... }

INPUT: COMMIT: Commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to writing files. COMMIT is the default.

ROLLBACK: reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely

linked to the waveform. (use `HELP RECALLTRACE` for a list of which header variables are committed, and which are rolled back.)

header: header field upon which to sort the files.

ASCEND: Sort files on header in ascending order This is the default.

DECEND: Sort files on header in descending order

DESCRIPTION: Sort the files in memory in order according to the header field given. The earlier a header field appears on the command line, the higher priority that field will receive in the sort, the first field receiving the highest priority, and subsequent fields used to break ties. No more than five header fields may be entered. Each may be followed by either `ASCEND` or `DESCEND` to indicate the direction of the sort on that particular field. If neither `ASCEND` nor `DESCEND` is specified, `ASCEND` will be used by default. If `Sort` is called without specifying any header fields, it will sort on the fields specified in the previous call to `SORT`. If the first call to `SORT` is without any header fields, it will produce error 1379.

DEFAULTS: It is presumed that all sorts will be in ascending order unless `DESCEND` is specified on the command line.

ERROR MESSAGES: 301: Out of memory.

1379: No `SORT` parameters given

1380: Too many SORT parameters:

1381: Not a valid SORT parameter:

1383: `SORT` failed

WARNING MESSAGES: 1384:

SEE COMMANDS: `COMMIT`, `ROLLBACK`, `RECALLTRACE`

LATEST REVISION: October 27, 1998 (Version 0.58)

3.2.23 Synchronize

SUMMARY: Synchronizes the reference times of all files in memory.

SYNTAX: `SYNCHRONIZE {ROUND {ON—OFF}} {BEGIN {ON—OFF}}`

INPUT: ROUND {ON} : Turn begin time rounding on. When this options is on, the begin times for each file are rounded to the nearest multiple of the sampling interval.

ROUND OFF : Turn begin time rounding off.

BEGIN {ON} : Sets begin time of each file to zero.

BEGIN OFF : Maintains the GMT values of the reference times.

DEFAULT VALUES: SYNCHRONIZE ROUND OFF BEGIN OFF

DESCRIPTION: This command synchronizes the reference times for all files in memory. It determines the latest starting time of all files by examining their reference times and beginning offset times. This latest starting time then becomes the reference time for ALL of the files in memory. New values for all of the offset times (B, E, A, Tn, etc.) for each of the files are then calculated. This command is useful when a set of files have different reference times and you want to use the CUT or XLIM command to analyze or plot portions of them. Once they have been synchronized to the same reference time, the cuts will then refer to the exact same GMT time window. If the BEGIN option is used, GMT values of reference times are not preserved. The BEGIN option sets the kztime of all files the same, it sets the kzdate of all files the same, and it sets the begin time of all files to zero. Other reference points retain their relation to the begin time of the file.

EXAMPLES: Assume you read two files into memory with different reference times:

```
u: READ FILE1 FILE2
u: LISTHDR B KZTIME KZDATE
s:
s: FILE: FILE1
s: _____
s: B = 0.
s: KZTIME = 10:38:14.000
s: KZDATE = MAR 29 (088), 1981
s:
s: FILE: FILE2
s: _____
s: B = 10.00
s: KZTIME = 10:40:10.000
s: KZDATE = MAR 29 (088), 1981
```

The files have the same reference date but different reference times and different beginning offsets. Now if you execute the SYNCHRONIZE command **followed by another LISTHDR you would find:**

```

u: SYNCHRONIZE
u: LISTHDR
s:
s: FILE: FILE1
s: —————
s: B = -126.00
s: KZTIME = 10:40:20.000
s: KZDATE = MAR 29 (088), 1981
s:
s: FILE: FILE2
s: —————
s: B = 0.
s: KZTIME = 10:40:20.000
s: KZDATE = MAR 29 (088), 1981

```

Now the files in memory have the same reference time which is the beginning of the later file. If there had been any defined time markers in these headers, their values would be adjusted so that they point to the same time as before.

HEADER CHANGES: NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, NZMSEC, B, E, A, O, Tn.

LATEST REVISION: May 15, 1987 (Version 10.2)

3.2.24 Wild

SUMMARY: Sets wildcard characters used in read commands to expand filelists.

SYNTAX: WILD {ECHO {ON—OFF}}, {SINGLE char}, {MULTIPLE char}, {CONCATENATION chars}

INPUT: ECHO {ON} : Turn echoing of expanded filelist on. Echoing is only when this option is on and there are wildcard characters in the

filelist.

ECHO OFF : Turn echoing of expanded filelist off.

SINGLE char : Change the character used to match single characters.

MULTIPLE char : Change the character used to match multiple characters.

CONCATENATION chars : Change the two characters used to enclose concatenation lists.

DEFAULT VALUES:

OPTION UNIX VAX PRIME ECHO ON ON ON SINGLE ? ? + MULTIPLE * * \ CONCATENATION [, \] (, \) [, \]

DESCRIPTION: This feature is available at the command level of many modern operating systems and is called “wildcarding” or “filename expansion.” It is a notation that allows you to abbreviate filenames and to specify entire groups of files using a simple shorthand notation. SAC has implemented wildcarding, along with several extensions, in its READ, READALPHA, and READHDR commands. Using **this notation, you can easily access lists such as:**

- All files beginning with the letters “abc”.
- All files ending with the letter “z”.
- All files with exactly three letters in their names.

There are three elements in this wildcard notation. We will use the default wildcard characters for the UNIX version in this description and in the examples below. The defaults may be different on the computer you are using. You may also use this command to change the characters to be anything you want. The multiple match character (“*”) is used to match an arbitrary character string, including an empty string. The single match character (“?”) is used to match any single character. The concatenation characters (“[” and “]”) are used to enclose a comma delimited list of character strings to match. The character strings in a concatenation list may not contain the single or multiple match wildcard characters. These are the steps that SAC uses to perform this wildcard filename **expansion:**

(1) Strip away the directory part of the token if it exists. Otherwise use the current directory.

- (2) Make a system call to get a list of all files in the directory.
- (3) If a concatenation list is in the token, form new tokens from each character string in the concatenation list with the other characters in the token and then match them to the list of files. If there is no concatenation list in the token, simply match the token to the list of files.
- (4) Remove any duplicate matches to form the expanded filelist.
- (5) Echo the expanded filelist if requested.
- (6) Attempt to read the expanded filelist into memory.

Each operating system uses a somewhat different scheme to store and access files in a directory. The system call in (1) above reflects these differences. For example, the filenames are returned in alphabetical order in UNIX but are not on the PRIME or VAX. The order of the files in a PRIME directory is arbitrary. These differences are reflected in the order of the files in the expanded filelist. You may have to experiment with different variations of wildcard characters and concatenation lists if the order of the files in the expanded list is important. The examples below will help clarify how to use these wildcard elements. One useful feature is that SAC saves the character strings contained in the concatenation list. When you enter an empty list, then the previous list is reused. This can save a lot of typing.

EXAMPLES: Assume that the contents of the current directory contain the following files **in the order shown**:

ABC
DEF
STA01E
STA01N
STA01Z
STA02E
STA02N
STA02Z
STA03Z

Also, assume that expanded filelist echoing is on. The following shows how the various wildcarding elements can be used to read parts of the above filelist into memory.

u: READ S\
s: STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
u: READ \
s: STA01Z STA02Z STA03Z
u: READ ???
s: ABC DEF
u: READ STA01[Z,N,E]
s: STA01Z STA01N STA01E
u: READ *[Z,N,E]
s: STA01Z STA02Z STA03Z STA01N STA02N STA01E STA02E
u: READ *1[Z,N,E] *2[]
s: STA01Z STA01N STA01E STA02Z STA02N STA02E

LIMITATIONS: You may have only one concatenation string in a token. This limitation will be eliminated in a future version. Several other wildcard and filename expansion options will also be added at that time.

SEE COMMANDS: READ, READALPHA, READHDR

LATEST REVISION: May 15, 1987 (Version 10.2)

3.2.25 Write

SUMMARY: Writes data in memory to disk.

SYNTAX: WRITE {options} {namingoptions} **where options are one or more of the following:**

SAC—ALPHA—XDR COMMIT—ROLLBACK—RECALLTRACE DIR
ON—OFF—CURRENT—name KSTCMP

These options MUST precede any element in the namingoptions:

OVER APPEND text PREPEND text DELETE text CHANGE text1
text2 filelist

Only one of these namingoptions is allowed at a time.

INPUT: no arguments : Use previous format and previous write filelist.

SAC : Write in SAC binary data file format.

ALPHA : Write in SAC alphanumeric data file format.

SEGY : Write file formatted according to the IRIS/PASSCAL form of the SEG Y format. This format allows one waveform per file. **Note:** only evenly-spaced, time-series files will be written in SEG Y.

The SCALE field in SAC is ignored. Since SAC stores the waveform as a series of floating point (real) numbers, and SEG Y stores the waveform as a series of long integers, the datapoints from SAC are normalized to the maximum allowable integer. The scale field in SEG Y is determined to be the factor which will restore the waveform as close as possible to that of the original SAC file, when read with the READ SEG Y command.

The following SAC header fields are saved as the **following SEG Y header fields:** SAC SEG Y — — KZDATE year, day, hour, minute, second, and m_secs are ... KZTIME set to BEGIN time corrected by KZDATE and KZTIME. BEGIN trigyear, trigday, trighour, trigminute, trigsecond, ORIGIN and trigmills are ORIGIN corrected by KZDATE & TIME.

NPTS sampleLength and/or num_samps DELTA deltaSample and/or samp_rate DEPMAZ max, corrected by SEG Y's scale. DEPMIN min, corrected by SEG Y's scale. DIST sourceToRecDist STLA recLatOrY (written as latitude in degrees) STLO recLongOrX (written as longitude in degrees) EVLA sourceLatOrY (written as latitude in degrees) EVLO sourceLongOrX (written as longitude in degrees) lats and lons are multiplied by 3600 to correct units STEL recElevation EVEL sourceSurfaceElevation EVDP sourceDepth

KSTNM station_name KCMPNM channel_name KEVNM event_number (only if KEVNM is numeric and j 1e09)

The following SEG Y fields are hardwired: SEG Y Value — — elevationScale 1 coordScale 1 coordUnits 2 gainType 1 gainConst 1 data_form 1

Our thanks to Steve Roecker or RPI for providing SAC2SEG Y which served as our starting point.

XDR : Write in SAC binary xdr format. This format is used for the moving binary data files to/from a different architecture, such as a pc running LINUX.

COMMIT : Commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to writing files. COMMIT is the default.

ROLLBACK : reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE : - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR ON : Turn directory option on but don't change name of write directory.

DIR OFF : Turn directory option off. When off, writes to current directory.

DIR CURRENT : Turn directory option on and set name of write directory to the “current directory” (e.g. the directory from which you started SAC.)

DIR name : Turn directory option on and set name of write directory to name. Write all filenames to the directory called name. This may be a relative or absolute directory name.

KSTCMP : Use the KSTNM and KCMPNM header variables to define a file name for each data file in memory. The names generated will be checked for uniqueness, and will have sequencing digits added as necessary to avoid name clashes.

OVER : Use current read filelist as write filelist. Overwrite files on disk with data in memory.

APPEND text : Write filelist is created by appending text to each name in the current read filelist.

PREPEND text : Write filelist is created by prepending text to each name in the current read filelist.

DELETE text : Write filelist is created by deleting the first occurrence of text in each name in the current read filelist.

CHANGE text1 text2 : Write filelist is created by changing the first

occurrence of text1 in each name in the current read filelist to text2.

filelist : Write filelist is set to filelist. This list may contain simple filenames, relative pathnames, or full pathnames. IT MAY NOT CONTAIN WILDCARDS.

DEFAULT VALUES: WRITE SAC COMMIT

DESCRIPTION: This command allows you, at any point in the processing of data, to save the results on disk. Several disk file formats are available. More will be added as needed. Each file in memory is written without being cut or desampled. Most of the time, you will want to use to the SAC data file format. This is a compact binary file format which is fast to read and write. It contains a large header record and one or two data records. See the Users Manual for details on the physical format. The alphanumeric data file format is an ASCII equivalent of the SAC data file format. It takes up much more room on disk and is much slower to read and write. It is useful if you wish to look at the content of the file using a text editor or wish to transfer data to a different kind of computer. You can directly specify the names of the files to write or you can indirectly specify them by having SAC modify the names of files that are currently in memory. The OVER options sets the write file list to the read file list. It is used to overwrite the last set of disk files read with the data that is currently in memory. The APPEND, PREPEND, DELETE, or CHANGE options create a write file list by modifying each of the names in the read file list in the requested way. This is very useful in macros where you are automatically processing large numbers of data files and need to keep trace of the output files in a consistent manner. The write file list is output when any of these four options is selected. This lets you see the names that were actually used.

EXAMPLES: To filter a set of data files and then save the results in a new set of data **files:**

u: READ D1 D2 D3

u: LOWPASS BUTTER NPOLES 4

u: WRITE F1 F2 F3

This could have also been done using the CHANGE option:

u: READ D1 D2 D3
u: LOWPASS BUTTER NPOLES 4
u: WRITE CHANGE D F
s: F1 F2 F3

Notice that SAC output the write file list in this case. To replace the original data on disk with the filtered data the third line in the above **example would be:**

u: WRITE OVER

Note: for examples of the behavior of COMMIT, ROLLBACK, and RECALLTRACE, see the commands of the same name.

ERROR MESSAGES: 1301: No data files read in.

1311: No list of filenames to write.

1312: Bad number of files in write file list:

- the number of files in the write file list must be the same as the number in the data file list (the number read into memory). **1303:** Overwrite flag is not on for file

- header variable LOVROK is .FALSE. - this provides some protection for valuable data.

SEE COMMANDS: READ, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.2.26 Writecss

SUMMARY: Writes data in memory to disk in CSS 3.0 format.

SYNTAX: WRITE {BINARY—ASCII} {COMMIT—ROLLBACK—RECALLTRACE}
 {DIR ON—OFF—CURRENT—name} name

INPUT: ASCII: (Default) Write standard ASCII flatfiles.

BINARY: Write output as a single CSS 3.0 binary file.

COMMIT : The COMMIT option commits headers and waveforms in SAC memory prior to writing the traces. COMMIT is the default.

ROLLBACK: The ROLLBACK option reverts to the last committed version of the header and waveform before writing the traces.

RECALLTRACE : The RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform before writing the traces. (use `HELP RECALLTRACE` for a list of which header variables are committed, and which are rolled back.)

DIR ON : Turn directory option on but don't change name of write directory.

DIR OFF : Turn directory option off. When off, writes to current directory.

DIR CURRENT : Turn directory option on and set name of write directory to the "current directory" (e.g. the directory from which you started SAC.)

DIR name : Turn directory option on and set name of write directory to name. Write all files to the directory called name. This may be a relative or absolute directory name.

name : Write filelist in set to name. There should be only one name specified. It may not contain wildcards. For ASCII output, name will be prepended to the table name for each flatfile. (i.e. name.wfdisc, name.origin, ...). In BINARY mode, name is the output file name.

DEFAULT VALUES: WRITECSS ASCII COMMIT DIR OFF

DESCRIPTION:

This command allows you, at any point in the processing of data, to save the results to disk in CSS 3.0 format. In ASCII mode (default) one or more ASCII flatfiles are written. The exact files written will depend **upon the source of the data but can be any of:**

wfdisc, wftag, origin, arrival, assoc, sitechan, site, affiliation, origerr, origin, event, sensor, instrument, gregion, stassoc, remark sacdata.

In Binary mode a single file will be written containing the same set of tables as would be written in ASCII mode, but with all tables in binary format and with the waveform data embedded in the file.

For more information on the CSS format see the "Center for Seismic Studies **Version 3 Database:** Schema Reference Manual".

ERROR MESSAGES: 1301: No data files read in.

1311: No list of filenames to write.

1312: Bad number of files in write file list

SEE COMMANDS: READ, READCSS, WRITE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: October 27, 1998 (Version 00.58)

3.2.27 Writegse

SUMMARY: Write data files in GSE 2.0 format from memory to disk.

SYNTAX: WRITEGSE {TYPE} {SOURCE ON—OFF—str} {COMMIT—ROLLBACK—RECALLTRACE} {DIR name} filename

INPUT: TYPE : Determines whether the data are written in ascii integer format (INT) or as compressed gse format (CM6). Default is INT.

SOURCE str : str is a string 20 characters or less specifying the institution at which the GSE file was written. str is written in the MSG_ID line of the resultant GSE file.

COMMIT : Commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : Reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR name : The directory in which to write the gsefile. This directory name is the same one that is used in WRITE command.

filename : The name of the gse file to be written.

DEFAULT VALUES: WRITEGSE INT SOURCE OFF COMMIT

DESCRIPTION: Writes all data in memory to a single file according to the GSE 2.0 data format

The following GSE Data messages are written: WAVEFORM
STATION CHANNEL ARRIVAL ORIGIN

Waveforms are written in INT format: floating point data is truncated to the nearest integer.

Note: There is no way in GSE 2.0 to associate ORIGIN data with a waveform, so SAC's READGSE command does not read ORIGIN data, but WRITEGSE writes it.

Note: SAC does not currently read nor write DETECTIONS information. Therefore, ARRIVAL information is not associated with specific channels.

LATEST REVISION: April 22, 1999 (Version 00.58)

3.2.28 Writehdr

SUMMARY: Overwrites the headers on disk with those in memory.

SYNTAX: WRITEHDR {COMMIT—ROLLBACK—RECALLTRACE}

INPUT: COMMIT : Commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to writing files. COMMIT is the default.

ROLLBACK : reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE : - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DEFAULT VALUES: WRITEHDR COMMIT

DESCRIPTION: The data on disk is NOT overwritten by this command. Use the WRITE OVER command to overwrite headers and data. The WRITEHDR command should NEVER be used if the CUT option is on. The header in memory is modified to reflect the effects of the CUT, but the data on disk is not modified. Use of the WRITEHDR command on cut data files will have the effect of apparently shifting and truncating the data

on disk in time.

ERROR MESSAGES: 1301: No data files read in.

HEADER CHANGES: Updates headers on disk.

LIMITATIONS: See description above about use of CUT and WRITE-HDR.

SEE COMMANDS: CUT, WRITE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.2.29 Writesdd

SUMMARY: Writes data in memory to disk in SDD format.

SYNTAX: WRITESDD {options} {namingoptions} **where options are one or more of the following:**

DIR ON—OFF—CURRENT—name

These options MUST precede any element in the namingoptions:

OVER APPEND text PREPEND text DELETE text CHANGE text1
text2 filelist

Only one of these namingoptions is allowed at a time.

INPUT: DIR ON : Turn directory option on but don't change name of write directory.

DIR OFF : Turn directory option off. When off, write files to current directory.

DIR CURRENT : Turn directory option on and set name of write directory to the "current directory" (e.g. the directory from which you started SAC.)

DIR name : Turn directory option on and set name of write directory to name. Write all filenames to the directory called name. This may be a relative or absolute directory name.

OVER : Use current read filelist as write filelist. Overwrite files on disk with data in memory.

APPEND text : Write filelist is created by appending text to each name in the current read filelist.

PREPEND text : Write filelist is created by prepending text to each name in the current read filelist.

DELETE text : Write filelist is created by deleting the first occurrence of text in each name in the current read filelist.

CHANGE text1 text2 : Write filelist is created by changing the first occurrence of text1 in each name in the current read filelist to text2.

filelist : Write filelist is set to filelist. This list may contain simple filenames, relative pathnames, or full pathnames. IT MAY NOT CONTAIN WILDCARDS.

LATEST REVISION: September 08, 1990 (Version 10.6)

3.2.30 Writetable

3.3 GCM: Graphics Control Module

3.3.1 Begindevices

SUMMARY: Begins plotting to one or more graphics devices.

SYNTAX: BEGINDEVICES devices

where devices is one or more of the following:

SGF, XWINDOWS, SUNWINDOWS

ALTERNATE FORMS: BEGG and BG are obsolete but acceptable names for this command.

INPUT: SGF : The SAC Graphics File device driver.

XWINDOWS : The X-windows window display system.

SUNWINDOWS : The Sun window display system.

DESCRIPTION: The arguments to this command consists of a list of one or more graphics devices. Subsequent plots are sent to the devices in this list. This remains in effect until the next execution of a BEGINDEVICES or ENDDEVICES command or until SAC is terminated. Details about each graphics device are given below. There are three graphics “devices” currently being supported. The first one, SAC Graphics File (SGF), is

a general purpose device drivers representing a large class of actual physical devices. The second, XWINDOWS, is a general windowing system running on most high-resolution, bit-mapped graphics workstations. The third, SUNWINDOW, is a windowing system running under SunOS 4.x. Each device is described in more detail below. The number, type, and names of the graphics devices available on your system may be different from this list. Check with your system administrator or use the REPORT DEVICES command to determine which devices are available. SGF stands for SAC Graphics File. A SAC Graphics File contains all the information needed to generate a single plot on any graphics device. (Using the current computer jargon, these are called graphics “metafiles.”) Each plot is stored in a separate file. The file names are of the form “Fnnn.SGF” where “nnn” is the plot number, beginning with “001”. You can control some features of this file name using the SGF command. Programs are available which can display these files to the terminal, merge several files into a single file, or produce an alphanumeric dump of a file for debugging. Programs are also available to convert these files to specific graphics devices such as the Apple Laserwriter, Houston Instruments pen plotter, etc. It is fairly easy to create one of these conversion programs. The format is described in the Users Manual. XWINDOWS (or X for short) is a windowing scheme developed under the industry-financed Athena project at MIT. X employs what is called a network model, where a single process or server controls the screen display. Other programs send requests to this server when they want to modify part of the screen. X is widely accepted in the graphics workstation area and seems to currently offer the best framework for developing portable window-based applications. SUNWINDOWS is a proprietary windowing system available on the workstations marketed by Sun Microsystems running SunOS 4.x. It is not based upon the network model. Each program contains its own set of procedures to control the screen. The two windowing systems are incompatible. You must be running under one or the other at any given time. Benchmarks using the SAC graphics library do not show significant difference in graphics display speeds between these two windowing systems.

(1) For an overview of X, there is an article by Peter A. Hack called “The Advantages of X” in the August 1987 issue of Computer Graphics World.

(2) **A technical reference on X is:** James Gettys, Ron Newman, and Tony Della Fera, “Xlib - C Language X Interface Protocol Version 10”, MIT (1986).

(3) For information about SUNWINDOWS, there are two manuals written by **Sun Microsystems**: “Windows and Window Based Tools”, and “Sun View Programmers’s Guide.”

SEE COMMANDS: ENDDEVICES, SGF

LATEST REVISION: Dec 23, 1997 (Version 0.56a)

3.3.2 Enddevices

SUMMARY: Terminates one or more graphics devices.

SYNTAX: ENDDEVICES TERMINAL,SGF,XWINDOWS,SUNWINDOWS

ALTERNATE FORMS: ENDG or EG are obsolete but acceptable names for this command.

INPUT: TERMINAL : The graphics terminal you are currently logged into.

SGF : The SAC Graphics File device driver.

XWINDOWS : The X-windows window display system.

SUNWINDOWS : The Sun window display system.

DESCRIPTION: This command terminates one or more graphics devices. Devices are activated using the BEGINDEVICES command. The BEGINDEVICES command has a description of each of these graphics devices.

SEE COMMANDS: BEGINDEVICES

LATEST REVISION: April 13, 1987 (Version 10.1)

3.3.3 Erase

SUMMARY: Erases the graphics display area.

SYNTAX: ERASE

DESCRIPTION: This command works only if SAC knows what graphics device you are using. This is true only if you have already done some plotting. This command is necessary for the ADM terminal which does not have an erase screen key and is useful in command files when you want the screen erased prior to sending out a large amount of text.

LATEST REVISION: October 11, 1984 (Version 9.1)

3.3.4 Hcd

3.3.5 Vspace

SUMMARY: Changes the maximum size and shape of plots.

SYNTAX: VSPACE FULL—*v*

INPUT: FULL : Use full viewspace. This is the largest possible screen or window size.

***v* :** Force the viewspace to have a *y*:*x* aspect ratio of *v*. The largest possible area with this aspect ratio becomes the viewspace.

DEFAULT VALUES: VSPACE FULL

DESCRIPTION: The viewspace represents that portion of the viewing surface on which plots can be drawn. There is a large variation in viewspace shapes and sizes **between different graphics devices:**

(1) Although differing greatly in size, many graphics terminals have an aspect ratio of 0.75. Some terminals, however, have different aspect ratios. The HP 26xx family of terminals have an aspect ratio of 0.5. The Tektronix 4025 terminals can have a wide range of aspect ratios, depending upon how many lines on the screen are assigned to the alphanumeric display and how many to the graphic display.

(2) The SAC Graphics File (SGF) has an aspect ratio of 0.75 This is the approximate ratio of a standard sheet of 8.5 by 11 paper.

(3) The graphics windows created by the XWINDOWS or SUNWINDOWS device can have any aspect ratio you wish.

This variation among graphics devices can be a problem if you are need complete control over the size and shape of a plot. This command gives

you control over the shape of a plot by letting you select a fixed aspect ratio. (SAC does not currently give you much control over the size.) The default is to plot to the full viewspace. If you do select a fixed aspect ratio, then the viewspace becomes the largest enclosed area on the device with that aspect ratio. This command is useful when you are creating a figure on an interactive device using PLOTG that you eventually want to send to the SGF device. You should set the aspect ratio to 0.75 before doing any plotting. This will ensure that the figure will have the same in the SGF file as it is does on the interactive device. Another use is when you want a square viewspace independent of the graphics device. This is easily done by requesting an aspect ratio of 1.0.

LATEST REVISION: May 15, 1987 (Version 10.2)

3.4 GEM: Graphic Environment Module

3.4.1 Axes

SUMMARY: Controls the location of annotated axes.

SYNTAX: AXES ON—OFF—ONLY sides

where sides is the keyword:

ALL

or one or more of the following:

TOP,BOTTOM,RIGHT,LEFT

ALTERNATE FORMS: AXIS may be used for AXES. (Useful for grammarians only.)

INPUT: ON : Turn axes on for listed sides; others unchanged.

OFF : Turn axes off for listed sides; others unchanged.

ONLY : Turn axes on only for listed sides; others off.

ALL : All four axes.

TOP : X axis above plot.

BOTTOM : X axis below plot.

RIGHT : Y axis to right of plot.

LEFT : Y axis to left of plot.

DEFAULT VALUES: AXES ONLY BOTTOM LEFT

DESCRIPTION: Axes can be drawn on one or more of the four sides of a plot. Axes annotation is drawn using the division spacing set by the XDIV command. Tick mark labeling is controlled independently using the TICKS command.

EXAMPLES: To turn on the top axes and leave the others unchanged:

u: AXES ON TOP

To turn off all axes annotation:

u: AXES OFF ALL

To turn axes annoation on for the bottom side and off for the rest:

u: AXES ONLY BOTTOM

SEE COMMANDS: XDIV, TICKS

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.2 Beginframe

SUMMARY: Turns off automatic new frame actions between plots.

SYNTAX: BEGINFRAME {PRINT {pname} }

INPUT: PRINT {pname} : When PRINT is used with BEGINFRAME, it signals the associated call to ENDFRAME to print the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

ALTERNATE FORMS: BEGFR is an obsolete but allowable form of this command.

DESCRIPTION: A “new frame action” is defined as the clearing of the current graphics **display surface**. **Specifically it is:**

- the erasing of the screen for a graphics terminal.
- the closing of the current file for the SGF device driver.
- the erasing of the current graphics window on a multi-window workstation.
- the advancing of the film one frame for a film device.
- the movement of the paper to a new area on a pen plotter.

Normally SAC does a new frame action before each new plot (PLOT, PLOT1, etc.) SAC stops doing this new frame action when the BEGINFRAME command is executed. It resumes automatic framing when the ENDFRAME command is executed. Therefore, all plot commands executed between these two commands will have their output placed on the same frame. By changing the viewport (XVPORT, YVPORT) between plot commands, by changing some of the various plot options, and by reading in different sets of data files, fairly complicated plots with multiple images can be easily generated. See the example and figure below. You MUST execute the ENDFRAME command to discontinue this mode and to resume automatic framing between plots.

EXAMPLES: The plot that follows was generated using the set of commands shown below. Comments about the process are given in parenthesis.

```

u: CUT A -0.2 N 512 (set up cut and read file)
u: READ FILE1
u: BEGINFRAME (turn off automatic framing)
u: XVPORT .1 .9 (define viewport and options)
u: YVPORT .7 .9
u: TITLE 'SEISMIC TRACE'
u: FILEID OFF (turn off fileid and qdp option)
u: QDP OFF
u: PLOT (plot the trace)
u: FFT WMEAN (take transform of data)
u: XVPORT .1 .45 (second viewport and options)
u: YVPORT .15 .55
u: TITLE 'Amplitude Response (linlog)'
u: YLIM 1E-5 1
u: PLOTSP AM LINLOG (plot the amplitude)
u: XVPORT .55 .9 (third viewport and options)
u: TITLE 'Amplitude Response (loglog)'
u: XLIM 1 60
u: PLOTSP AM LOGLOG (plot amplitude again)
u: ENDFRAME (resume automatic framing)

```


u: CUT OFF (reset parameters used to default values)

u: FILEID ON

u: XLIM OFF

u: YLIM OFF

The last four commands reset some of the parameters used in this operation to their default values. This is a good habit to get into, especially when writing macros, as a way of avoiding the problem of one macro effecting the operation of others that follow.

SEE COMMANDS: ENDFRAME, XVPORT, YVPORT

LATEST REVISION: May 15, 1987 (Version 10.2) Use of BEGINFRAME and ENDFRAME to Create a Special Plot

3.4.3 Beginwindow

SUMMARY: Begins plotting to a new graphics window.

SYNTAX: BEGINWINDOW *n*

INPUT: *n* : The graphics window number to begin plotting in. There are a total of five graphics windows.

DEFAULT VALUES: BEGINWINDOW 1

DESCRIPTION: Many of the newer graphics terminals and workstations support the concept of multiple “windows”. Different jobs or activities can run in each window and display their results on the screen at the same time. “X-windows” and “Sun windows” are two of the more popular systems currently available. If you are using a device that supports one of these systems, then you can use multiple graphics windows in SAC to display your results. If you are not using such a device, SAC will accept but ignore all commands that refer to multiple graphics windows. There are two commands that control the use of this multi-windowing option. The WINDOW command lets you control the location and shape of the graphics windows. The BEGINWINDOW command lets you select the window in which to display subsequent plots. BEGINWINDOW will create the requested window if it does not currently exist on your display. The WINDOW command only works BEFORE the window is created. On most systems you can also move

and resize these windows dynamically using the mouse and pop-up menus. Generally but not always (you should check for yourself), the moving of a window will result in the current plot being automatically redrawn whereas the resizing of a window results in the current plot being redrawn but not rescaled. The next plot in a resized window will be scaled correctly. All text (the commands you type and SAC's responses) are displayed in the window in which you started SAC.

SEE COMMANDS: WINDOW

LATEST REVISION: May 15, 1987 (Version 10.2)

3.4.4 Border

SUMMARY: Controls the plotting of a border around plots.

SYNTAX: BORDER {ON—OFF}

INPUT: {ON} : Turn border plotting on.

OFF : Turn border plotting off.

DEFAULT VALUES: BORDER OFF

DESCRIPTION: When this option is on, a solid border is drawn around the sides of the plot at the edge of the viewport (see XVPORT.) Note that an axis line is always drawn on each side of the plot that contains an annotated axis (see AXES) or a set of tick marks (see TICKS). Thus the border option only applies to those sides without axes or tick marks.

SEE COMMANDS: XVPORT, YVPORT, AXES, TICKS

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.5 Color

SUMMARY: Controls color selection for color graphics devices.

SYNTAX: COLOR {ON—OFF—color} options **where options are one or more of the following:**

{INCREMENT {ON—OFF}} {SKELETON color} {BACKGROUND color} {LIST STANDARD—colorlist}

and where color is one of the following:

WHITE—RED—GREEN—YELLOW—BLUE—MAGENTA—CYAN—BLACK

SPECIAL NOTE: The LIST option must appear last in this command.

INPUT: color : The name of a standard color or the number of a color from the color table.

COLOR ON : Turn color option on but don't change data color.

COLOR OFF : Turn color option off.

COLOR color : Change data color and turn color option on.

INCREMENT {ON} : Increment data color from color list after each data file is plotted.

INCREMENT OFF : Do not increment data color.

SKELETON color : Change color of skeleton to standard color name or color table number.

BACKGROUND color : Change background color to standard color name or color table number.

LIST colorlist : Change the content of the color list. Enter list of standard color names or color table numbers. Sets data color to first color in list and turns color option on.

LIST STANDARD : Change to the standard color list. Sets data color to first color in list and turns color option on.

DEFAULT VALUES: COLOR BLACK INCREMENT OFF SKELETON BLACK BACKGROUND WHITE LIST STANDARD

TYPE: Parameter-setting

FUNCTIONAL MODULE: Graphic Environment

DESCRIPTION: This command controls color attributes for those devices which can display a large number of colors. The data color is the color that is used when plotting the data files. The data color may be automatically incremented from a color list after each data file is plotted. The skeleton color is the color used to plot and label the axes, titles, grids, and frame ids. The background color is the color of an empty frame, before any lines or text are plotted. Most of the time you will select the name of a standard color, such as red or blue. This will be the color, independent of the selected graphics device. At times, however, you may want to choose a non-standard color, such as aquamarine. This can be done by "downloading" a color table to the graphics device. This color table associates a specific

hue, saturation, and lightness with a specific integer number. You can then select aquamarine for a particular part of the plot by setting that attribute to the correct number from the color table. This may sound like a lot of work, but if aquamarine is your favorite color, it may be worth it. If you are plotting several data files on the same plot, you may want each to be in a different color. This is done using the INCREMENT option. When this option is on, the data color is incremented from a list of colors each time a data file is plotted. The order of colors in the standard or **default color list is given below:**

RED, GREEN, BLUE, YELLOW, CYAN, MAGENTA, BLACK

You may change the order or content of this color list using the LIST option. This is useful if you are doing a series of overlay plots (see PLOT2) and want the same colors used in the same order on each plot.

EXAMPLES: To select an incrementing data color starting with red:

u: COLOR RED INCREMENT

To select red data colors on a white background with a blue skeleton:

u: COLOR RED BACKGROUND WHITE SKELETON BLUE

To set up an incrementing data color list of red, white, and blue with an **aquamarine (!!!) background:**

u: COLOR RED INCREMENT BACKGROUND 47 LIST RED WHITE BLUE

The above example assumes that aquamarine is color 47 in the color table for the selected graphics device. Background color is currently being ignored.

LATEST REVISION: April 13, 1987 (Version 10.1)

3.4.6 Colortable

3.4.7 Endframe

SUMMARY: Resumes automatic new frame actions between plots.

SYNTAX: ENDFRAME

ALTERNATE FORMS: ENDFR is an obsolete but allowable form of this command.

DESCRIPTION: See the BEGINFRAME documentation.

SEE COMMANDS: BEGINFRAME

LATEST REVISION: May 15, 1987 (Version 10.2)

3.4.8 Floor

SUMMARY: Puts a minimum value on logarithmically scaled data.

SYNTAX: FLOOR {ON—OFF—v}

INPUT: {ON} : Turn floor option on but don't change value of floor.

OFF : Turn floor option off.

v : Turn floor option on and change value of floor.

DEFAULT VALUES: FLOOR 1.0E-10

DESCRIPTION: The floor option applies only when logarithmic scaling is being used. It applies to both the x and y axes. When this option is on, any data values less than the floor are set to the floor before plotting. By using a small positive value for the floor, errors in taking logarithms of non-positive numbers are avoided.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.9 Grid

SUMMARY: Controls the plotting of grid lines in plots.

SYNTAX: GRID {ON—OFF—SOLID—DOTTED}

INPUT: ON : Turn grid plotting on but don't change grid type.

OFF : Turn grid plotting off.

SOLID : Turn grid plotting on using solid grid lines.

DOTTED : Turn grid plotting on using dotted grid lines.

DEFAULT VALUES: GRID OFF

FUNCTIONAL MODULE: Graphic Environment

DESCRIPTION: This command controls grid lines in both directions. The XGRID and YGRID commands can be used to generate grid lines in only one direction.

SEE COMMANDS: XGRID, YGRID

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.10 Gtext

SUMMARY: Controls the quality and font of text used in plots.

SYNTAX: GTEXT {SOFTWARE—HARDWARE},{FONT n},{SIZE size}

where size is one of the following:

TINY—SMALL—MEDIUM—LARGE

INPUT: SOFTWARE : Use software text in plots.

HARDWARE : Use hardware text in plots.

FONT n : Set software text font to n. The range for n is currently 1 to 8.

FORCE n : Use hardware text in all cases for plots. Overrides HARDWARE option. HARDWARE still uses software for rotated fonts.

SIZE size : Change default text size. See TSIZE command for definitions of text sizes.

DEFAULT VALUES: GTEXT SOFTWARE FONT 1 SIZE SMALL

DESCRIPTION: Software text uses the text display capabilities of the graphics library. Characters are stored as small line segments and thus can be scaled to any desired size and can be rotated to any desired angle. Use of software text will produce the same result on different graphics devices. Use of software text is slower than hardware text, especially to the terminal. There **are currently 8 software fonts available:** simplex block (font 1), simplex italics (2), duplex block (3), duplex italics (4), complex block (5), complex italics (6), triplex block (7), and triplex italics (8). Examples of each font and each default text size is shown in the figure on the next page. Hardware text uses the text display capabilities of the graphics device itself. Hardware text sizes vary considerably between devices, so its use can

produce different looking plots on different devices. If a device has more than one hardware text size, the one closest to the desired size is used. Its primary asset is that it is much faster than software text and should therefore be used only when speed is more important than quality.

EXAMPLES: To select the triplex software font:

u: GTEXT SOFTWARE FONT 6

SEE COMMANDS: TSIZE

LATEST REVISION: July 22, 1991 (Version 9.1) Text Fonts and Default Text Sizes

3.4.11 Line

SUMMARY: Controls the linestyle selection in plots.

SYNTAX: LINE {ON—OFF—SOLID—DOTTED—n} {INCREMENT {ON—OFF}}, {LIST STANDARD—nlist}

INPUT: {ON} : Turn line drawing on. Don't change linestyle.

OFF : Turn line drawing off.

SOLID : Change to solid linestyle and turn line drawing on.

DOTTED : Change to dotted linestyle and turn line drawing on.

n : Change to linestyle n and turn line drawing on. A linestyle of 0 is the same as turning line drawing off. The number of linestyles varies from one graphics device to another.

INCREMENT {ON} : Increment linestyle from linestyle list after each data file is plotted.

INCREMENT OFF : Do not increment data linestyle.

LIST nlist : Change the content of the linestyle list. Enter list of linestyle numbers.

LIST STANDARD : Change to the standard linestyle list.

DEFAULT VALUES: LINE SOLID INCREMENT OFF LIST STANDARD

DESCRIPTION: This command controls the linestyle used when plotting data. The skeleton (axes, titles, etc.) are always plotted using solid lines. Grid linestyle is controlled by the GRID command. Not all graphics

devices have more than the solid linestyle. This command obviously has no effect on those devices. Also linestyle n may not be the same from device to device. There are other commands that control other aspects of the data display. The SYMBOL command can be used to display a set of scalable, centered symbols at each data point. The COLOR command controls color selection for color graphics devices. All of these attributes are independent of each other. You may select a blue dotted line with a symbol at each data point if you desire. A linestyle of 0 is the same as turning line drawing off. This is useful in the LIST option and the SYMBOL command to display some data with lines and some with symbols on the same plot. See the example below.

EXAMPLES: To select an incrementing linestyle starting with linestyle 1:

u: LINE 1 INCREMENT

To change the linestyle list to contain linestyles 3, 5, and 1:

u: LINE LIST 3 5 1

To plot three files on the same plot using PLOT2 with the first file plotted using a solid line and no symbol, the second with no line and a triangle symbol, and the third with no line and a cross symbol:

u: READ FILE1 FILE2 FILE3

u: LINE LIST 1 0 0 INCREMENT

u: SYMBOL LIST 0 3 7 INCREMENT

u: PLOT2

SEE COMMANDS: SYMBOL, COLOR

LATEST REVISION: October 11, 1984 (Version 9.1)

3.4.12 Linlin

SUMMARY: Turns on linear scaling for the x and y axes.

SYNTAX: LINLIN

DEFAULT VALUES: Linear scaling for both axes.

SEE COMMANDS: LINLOG, LOGLOG, LOGLIN

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.13 Linlog

SUMMARY: Turns on linear scaling for x axis and logarithmic for y axis.

SYNTAX: LINLOG

DEFAULT VALUES: Linear scaling for both axes.

SEE COMMANDS: LINLIN, LOGLOG, LOGLIN

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.14 Loglab

SUMMARY: Controls labels on logarithmically scaled axes.

SYNTAX: LOGLAB {ON—OFF}

INPUT: {ON} : Turn log labeling option on.

OFF : Turn log labeling option off.

DEFAULT VALUES: LOGLAB ON

DESCRIPTION: Labels are normally put on each decade of logarithmically interpolated axes. Secondary labels (ones between full decades) are placed on these axes if this option is on and if there is enough room.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.15 Loglin

SUMMARY: Turns on logarithmic scaling for x axis and linear for y axis.

SYNTAX: LOGLIN

DEFAULT VALUES: Linear scaling for both axes.

SEE COMMANDS: LINLIN, LINLOG, LOGLOG

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.16 Loglog

SUMMARY: Turns on logarithmic scaling for the x and y axes.

SYNTAX: LOGLOG

DEFAULT VALUES: Linear scaling for both axes.

SEE COMMANDS: LINLIN, LINLOG, LOGLIN

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.17 Null

SUMMARY: Controls the plotting of null values.

SYNTAX: NULL {ON—OFF—value}

INPUT: **ON** : Turns the NULL option on for plotting.

OFF : Turns the NULL option off for plotting.

value : Sets the value of a NULL to be value.

DEFAULT VALUES: NULL OFF

DESCRIPTION: Many times in a data set, when there are gaps in the data, no data is available. In many cases the data has been set to a predefined value. Typical values are 0.0, -1.0, -99. Usually the user will not want these values displayed on plots. The NULL command allows the user to define the NULL value and not connect a line through these data points. To set the NULL value to -1.0 and enable the NULL option during plotting:

u: NULL ON -1.0

LATEST REVISION: March 20, 1992 (Version 10.6e)

3.4.18 Plabel

SUMMARY: Defines general plot labels and their attributes.

SYNTAX: PLABEL {n} {ON—OFF—text},{SIZE size}, {BELOW—POSITION
x y {a}}

where size is one of the following:

TINY—SMALL—MEDIUM—LARGE

INPUT: **n** : Plot label number. There is currently a maximum of five plot labels. If omitted, the previous label number is incremented by one.

ON : Turn this plot label on.

OFF : Turn this plot label off.

text : Change text of plot label. Also turns plot label on.

SIZE size : Change the plot label size.

TINY : Tiny text size has 132 characters per line.

SMALL : Small text size has 100 characters per line.

MEDIUM : Medium text size has 80 characters per line.

LARGE : Large text size has 50 characters per line.

BELOW : Position this label “below” the previous label.

POSITION x y a : Define a specific position for this label. The range of the positions are: 0. to 1. for x and 0. to the maximum viewspace (normally 0.75) for y. a is the angle of the label in degrees clockwise from horizontal.

DEFAULT VALUES: Default size is small. Default position for label 1 is 0.15 0.2 0. Default position for other labels is below previous label.

DESCRIPTION: This command lets you define general purpose plot labels for subsequent plot commands. You can define the location and size of each label. The text quality and font used can be set using the GTEXT command. You can also generate a title and axes labels using the TITLE, XLABEL, and YLABEL commands.

EXAMPLES: The following commands would generate a four line label in the upper left **hand corner of subsequent plots:**

u: PLABEL ‘Sample seismogram’ POSITION .12 .5

u: PLABEL ‘from earthquake’

u: PLABEL ‘on January 24, 1980’

u: PLABEL ‘in Livermore Valley, CA’

An additional tiny label could be placed in the lower left hand corner:

u: PLABEL 5 ‘LLNL station: CDV’ S T P .12 .12

SEE COMMANDS: GTEXT, TITLE, XLABEL, YLABEL

LATEST REVISION: July 22, 1991 (Version 9.1)

3.4.19 Qdp

SUMMARY: Controls the “quick and dirty plot” option.

SYNTAX: QDP {ON—OFF—n},{TERM ON—OFF—n},{SGF ON—OFF—n}

INPUT: ON : Turn QDP option on for both the terminal and SAC Graphics File (SGF) devices.

OFF : Turn QDP option off for both devices.

n : Turn QDP option on for both devices and change the approximate number of data points to plot to n.

TERM ON : Turn quick and dirty plotting on for the terminal.

TERM OFF : Turn quick and dirty plotting off for the terminal.

TERM n : Turn QDP option on for the terminal and change the approximate number of data points to plot to n.

SGF ON : Turn quick and dirty plotting on for the SGF.

SGF OFF : Turn quick and dirty plotting off for the SGF.

SGF n : Turn QDP option on for the SGF and change the approximate number of data points to plot to n.

DEFAULT VALUES: QDP TERM 5000 SGF 5000

DESCRIPTION: Plotting large files (greater than say 1000 points) can take a long time. The “quick and dirty plot” option speeds up plotting by NOT plotting each data point. When this option is on, SAC will compute a section size by dividing the number of data points in the file by the number of data points you want displayed. The larger the file, the more data points in each section. SAC then computes and displays only the minimum and the maximum data point in each section. SAC displays a “desampling factor” (half the section size) in a small box in the corner of the plot when this option is on. Displayed data points may be somewhat closer or further apart than this number indicates since the extremum in each region are being plotted. There is a separate QDP option for the terminal and the SAC Graphics File device. The terminal QDP factor also applies to the XWINDOWS and SUNWINDOWS graphics devices. By default the QDP factor is considerably smaller for the terminal than for the SGF. This allows for very fast plots to the terminal and a more representative plot to the SGF. If both devices are on at the same time, the terminal QDP option applies. You may turn either of these options off or change the number of displayed points.

EXAMPLES: Assume FILE1 has 2100 data points and FILE2 has 4700 data points. If you **typed:**

u: READ FILE1 FILE2

u: BEGINDEVICES TERMINAL

u: PLOT

both plots would contain approximately 200 data points. The plot of FILE1 would contain approximately every tenth data point and the plot of FILE2 every twenty-third data point. The section size is rounded down to ensure that you will see at least the number of requested data points. If you now **plotted those same files to the SGF:**

u: BEGINDEVICES SGF

u: PLOT

both plots would contain approximately 1000 data points. If both devices were on, the plots would contain approximately 200 data points, the factor for the terminal.

LATEST REVISION: February 20, 1985 (Version 9.13)

3.4.20 Symbol

SUMMARY: Controls the symbol plotting attributes.

SYNTAX: SYMBOL {ON—OFF—n} {SIZE v},{SPACING v}, {INCREMENT {ON—OFF}}, {LIST STANDARD—nlist}

INPUT: ON : Turn symbol plotting on. Don't change symbol number.

OFF : Turn symbol plotting off.

n : Turn symbol plotting on. Change symbol number to n. There are 16 different symbols. A symbol number of 0 is the same as turning symbol plotting off.

SIZE v : Set symbol size to v. A value of 0.01 sets the size to 1 percent of the full plot size.

SPACING v : Set symbol spacing to v. This is the minimum spacing between drawn symbols. Use 0 if you want a symbol at every data point. Use 0.2 to 0.4 for annotating lines.

INCREMENT {ON} : Increment symbol number after each data file. The symbol number is the next one in the symbol list.

INCREMENT OFF : Do not increment symbol number.

LIST nlist : Change the content of the symbol list. Enter list of

symbol numbers. Sets symbol number to first entry in list and turns symbol plotting on.

LIST STANDARD : Change to the standard symbol list. Sets symbol number to first entry in list and turns symbol plotting on.

DEFAULT VALUES: SYMBOL OFF SIZE 0.01 SPACING 0. INCREMENT OFF LIST STANDARD

DESCRIPTION: The figure that follows shows each of the sixteen symbols. Symbol 1 cannot be scaled in size. It is a replacement for the point symbol which does not show up well on many devices (e.g. Versatec, pen plotter). This figure also shows examples of different symbol size and spacing values. These symbol attributes are independent of the line drawing attributes defined by the LINE command. With line drawing on, they can be used to annotate different lines on the same plot. By turning the line drawing off, they can be used to create scatter plots. If you are plotting several data files on the same plot, you may want each to be plotted with a different symbol. This is done using the INCREMENT option. When this option is on, the symbol is incremented from a list of symbols each time a data file is plotted. The default symbol list contains symbols 2 through 16. You may change the order or content of this list using the LIST option. This is useful if you are doing a series of overlay plots (see PLOT2) and want the same symbols used in the same order on each plot. A symbol number of 0 is the same as turning symbol plotting off. This is useful in the LIST option and the LINE command to display some data with lines and some with symbols on the same plot. See the example below.

EXAMPLES: To create a scatter plot, turn the line drawing off, choose an appropriate **symbol, and plot:**

u: LINE OFF

u: SYMBOL 5

u: PLOT

To annotate four solid lines on a PLOT2 plot using symbols 7, 4, 6, and 8, **and a spacing of 0.3:**

u: LINE SOLID

u: SYM SPACING .3 INCREMENT LIST 7 4 6 8

u: R FILE1 FILE2 FILE3 FILE4

u: PLOT2

To plot three files on the same plot using PLOT2 with the first file plotted using a solid line and no symbol, the second with no line and a triangle symbol, and the third with no line and a cross symbol:

u: READ FILE1 FILE2 FILE3

u: LINE LIST 1 0 0 INCREMENT

u: SYMBOL LIST 0 3 7 INCREMENT

u: PLOT2

SEE COMMANDS: LINE

LATEST REVISION: October 11, 1984 (Version 9.1) Summary of SYMBOL Command Attributes

3.4.21 Ticks

SUMMARY: Controls the location of tick marks on plots.

SYNTAX: TICKS ON—OFF—ONLY sides

where sides is the keyword:

ALL

or one or more of the following:

TOP,BOTTOM,RIGHT,LEFT

INPUT: ON : Turn ticks on for listed sides; others unchanged.

OFF : Turn ticks off for listed sides; others unchanged.

ONLY : Turn ticks on only for listed sides; others off.

ALL : All four ticks.

TOP : X axis above viewport.

BOTTOM : X axis below viewport.

RIGHT : Y axis to right of viewport.

LEFT : Y axis to left of viewport.

DEFAULT VALUES: TICKS ON ALL

DESCRIPTION: Tick marks can be drawn on one or more of the four sides of a plot. They are drawn at the current division spacing controlled by

the XDIV command. Tick marks are automatically drawn on sides where annotated axes have been requested using the AXES command.

EXAMPLES: To turn on the top tick marks and leave the others unchanged:

u: TICKS ON TOP

To turn off all tick marks (at least where there are no annotated axes):

u: TICKS OFF ALL

To turn tick marks on for the bottom side and off for the rest:

u: TICKS ONLY BOTTOM

SEE COMMANDS: XDIV, AXES

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.22 Title

SUMMARY: Defines the plot title and attributes.

SYNTAX: TITLE {ON—OFF—text},{LOCATION location},{SIZE size} **where location is one of the following:**

TOP—BOTTOM—RIGHT—LEFT

and where size is one of the following:

TINY—SMALL—MEDIUM—LARGE

INPUT: ON : Turn title option on. Don't change title text.

OFF : Turn title option off.

text : Turn title option on. Change text of title. If text contains embedded blanks, it must be enclosed in single quotes.

LOCATION location : Change location of title.

TOP : Top of the plot window.

BOTTOM : Bottom of the plot window.

RIGHT : To the right of the plot window.

LEFT : To the left of the plot window.

SIZE size : Change title text size.

TINY : Tiny text size has 132 characters per line.

SMALL : Small text size has 100 characters per line.

MEDIUM : Medium text size has 80 characters per line.

LARGE : Large text size has 50 characters per line.

DEFAULT VALUES: TITLE OFF LOCATION TOP SIZE SMALL

DESCRIPTION: If this option is on, a title is placed on each plot.

The size and location of the title can be changed as well as the text of the title itself. The text quality and font used can be set using the GTEXT command.

SEE COMMANDS: GTEXT

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.23 Tsize

SUMMARY: Controls the text size attributes.

SYNTAX: TSIZE {size v},{RATIO v},{OLD—NEW} **where size is one of the following:** TINY—SMALL—MEDIUM—LARGE

INPUT: size v : Change the value of one of the text sizes to v.

RATIO v : Change the text width to height ratio to v.

OLD : Change the values of all the text sizes to their “old” values.

These are the values used by SAC prior to Version 9.

NEW : Change the values of all the text sizes to their “new” values.

These are the default values when SAC is initialized.

DEFAULT VALUES: TSIZE RATIO 1.0 NEW

DESCRIPTION: Most of the text annotation commands (TITLE, XLABEL, FILEID, etc.) allow you to change the size of the text being displayed. You may choose from a set of four named sizes (TINY, SMALL, MEDIUM, and LARGE.) Each named size has an initial value given in the table below. These sizes are the height of a character as a fraction of the full (0.0 to 1.0) viewspace. There are times when you may want some of this annotation to be of a size different from these default values. TSIZE allows you to redefine any or all of these four named sizes. You may also use this command to change the width to height ratio of the characters. The default text sizes were changed, starting with Version 9 of SAC. The new set covers a wider range and generally looks better on most devices. You can easily

change back to the original set of sizes by using the OLD option. This might be useful if you want to create a plot that looks very similar to one that was generated using an older version of SAC. Also old PLOT files and macros will not look the same when replotted unless you first set the text sizes to their old values. The NEW option resets the sizes to their default values.

DEFAULT TEXT SIZES

NAME A B C D E TINY 0.015 66 50 68 110 SMALL 0.020 50 37 66 82
MEDIUM 0.030 33 25 44 55 LARGE 0.040 25 18 33 41

The column definitions in the table above are as follows:

A Height of character as a fraction of full viewspace.

B Number of lines of text in full viewspace.

C Number of lines of text in a normal viewspace. Normal means 0. to 1. in x and 0. to 0.75 in y.

D Minimum number of characters per normal viewspace line.

E Average number of characters per normal viewspace line. This is larger because the text is proportionally spaced.

EXAMPLES: To change the definition of MEDIUM and then use it to create a specially **sized title**:

u: TSIZE MEDIUM 0.35

u: TITLE 'Rayleigh Wave Spectra' SIZE MEDIUM

u: PLOT2

To reset this (and any other) size definitions to their default values:

u: TSIZE NEW

SEE COMMANDS: TITLE, XLABEL, FILEID, PLOT

LATEST REVISION: July 22, 1991 (Version 9.1)

3.4.24 Wait

SUMMARY: Tells SAC whether or not to pause between plots.

SYNTAX: WAIT {ON—OFF—EVERY}

INPUT: {ON} : Turn wait option on in normal mode.

OFF : Turn wait option off.

EVERY : Turn wait option on in every plot mode.

DEFAULT VALUES: WAIT ON

DESCRIPTION: When you read in more than one data file and then plot them using the PLOT command, one frame is generated for each file. If you are plotting to the terminal, SAC normally pauses after each plot and sends the message “WAITING” to the terminal. You can then hit the return key to see the next plot, type “GO” to have SAC plot the rest of the current set of plots without pausing, or type “KILL” to terminate the plotting of this set of files. SAC does not pause after the last plot, because the normal input prompt serves the same function. When this wait option is off, SAC does not pause between plots. With the wait option in the “every plot” mode SAC will pause between every plot, not just the ones generated by the PLOT command. This is useful when you are running SAC under the control of a command file or job control program.

EXAMPLES: The following example shows how SAC functions in the normal wait mode:

u: READ FILE1 FILE2 FILE3 FILE4

u: PLOT

s: waiting {plot of FILE1 to terminal}

u: (return)

u: waiting {plot of FILE2}

u: kill {user has seen enough}

s: (prompt) {SAC now waiting for next command}

LATEST REVISION: October 11, 1984 (Version 9.1)

3.4.25 Width

SUMMARY: Controls line-width selection for graphics devices.

SYNTAX: WIDTH {ON—OFF—linewidth} options **where options are one or more of the following:**

{SKELETON width} {INCREMENT {ON—OFF}} {LIST STANDARD—widthlist}

and where linewidth, width and widthlist are integer values.

SPECIAL NOTE: The LIST option must appear last in this command.

INPUT: WIDTH ON : Turn WIDTH option on but don't change current width values.

WIDTH OFF : Turn width option off.

WIDTH linewidth : Change data width to linewidth and turn WIDTH option on.

SKELETON width : Change width of skeleton to width and turn WIDTH option on.

INCREMENT {ON} : Increment width from widthlist list after each data file is plotted.

INCREMENT OFF : Do not increment data line width.

LIST widthlist : Change the content of the width list. Enter list of widths. Sets data width to first width in list and turns width option on.

LIST STANDARD : Change to the standard width list. Sets data width to first width in list and turns width option on.

DEFAULT VALUES: WIDTH OFF SKELETON 1 INCREMENT OFF LIST STANDARD

TYPE: Parameter-setting

FUNCTIONAL MODULE: Graphic Environment

DESCRIPTION: This command controls width attributes for those devices which can display a large number of line-widths. The data width is the width that is used when plotting the data files. The data width may be automatically incremented from a width list after each data file is plotted. The skeleton width is the width used to plot the axes. Only plot axes change with SKELETON option. Grids, text, labels and frame ids are always displayed with the thin line-width of value 1. If plotting several data files on the same plot, you may want each to be in a different width. This is done using the INCREMENT option. When this option is on, the data width is incremented from a list of widths each time a data file is plotted. The order and value of widths in the standard (default) **list is:**

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

You may change the order or content of this list using the LIST option. This is useful for doing overlay plots (see PLOT2) when you want the same widths used in the same order on each plot.

EXAMPLES: To select an incrementing data width starting with 1:

u: WIDTH 1 INCREMENT

To set up an incrementing data width list of 1, 3, and 5 with an skeleton of 2:

u: WIDTH SKELETON 2 INCREMENT list 1 3 5

In Xwindows widths that are odd actually look a little odd.

LATEST REVISION: June 20, 1992 (Version 10.6e)

3.4.26 Window

SUMMARY: Sets the location and shape of graphics windows.

SYNTAX: WINDOW n {XSIZE xmin xmax} {YSIZE ymin ymax}

INPUT: n : The graphics window number of interest. There are a total of five graphics windows.

X xmin xmax : Set the x (horizontal) location of graphics window n on the screen. xmin is the location of the left edge of the window and xmax is the location of the right edge. The range of these screen coordinates is 0.0 to 1.0.

Y ymin ymax : Set the y (vertical) location of graphics window n on the screen. ymin is the location of the bottom edge of the window and ymax is the location of the top edge. The range of these screen coordinates is 0.0 to 1.0.

Default Graphics Window Locations

n xmin xmax ymin ymax 1 0.05 0.85 0.05 0.60 2 0.07 0.87 0.07 0.62 3 0.09 0.89 0.09 0.64 4 0.11 0.91 0.11 0.66 5 0.13 0.93 0.13 0.68

DESCRIPTION: Many of the newer graphics terminals and workstations support the concept of multiple “windows”. Different jobs or activities can run in each window and display their results on the screen at the same time. “X-windows” and “Sun windows” are two of the more popular systems currently available. If you are using a device that supports one of these systems, then you can use multiple graphics windows in SAC to display your

results. If you are not using such a device, SAC will accept but ignore all commands that refer to multiple graphics windows. There are two commands that control the use of this multi-windowing option. The WINDOW command lets you control the location and shape of the graphics windows. The BEGINWINDOW command lets you select the window in which to display subsequent plots. BEGINWINDOW will create the requested window if it does not currently exist on your display. The WINDOW command only works BEFORE the window is created. On most systems you can also move and resize these windows dynamically using the mouse and pop-up menus. Generally but not always (you should check for yourself), the moving of a window will result in the current plot being automatically redrawn whereas the resizing of a window results in the current plot being redrawn but not rescaled. The next plot in a resized window will be scaled correctly.

EXAMPLES: To set the y (vertical) location of window 3 without modifying the x (**horizontal**) location:

u: WINDOW 3 Y 0.1 0.9

LATEST REVISION: May 15, 1987 (Version 10.2) Graphics Window Location on a Screen

3.4.27 Xdiv

SUMMARY: Controls the x axis division spacing.

SYNTAX: XDIV {NICE—INCREMENT *v*—NUMBER *n*},{POWER {ON/OFF}}

INPUT: NICE : Use “nice-numbered” division spacings.

INCREMENT *v* : Set division spacing increment to *v*.

NUMBER *n* : Set number of division spacings to *n*.

POWER {ON} : Turn power option on. When this option is on, SAC may print the division spacings as a number raised to a power of 10.

POWER OFF : Turn power option off.

DEFAULT VALUES: XDIV NICE POWER ON

DESCRIPTION: This command controls the selection of x axis division spacings. Most of the time the default “nice-numbered” spacings are

satisfactory. SAC determines these based on the minimum and maximum axis limits, the length of the axis, and the current axis character size. You may also force the division spacing to be a certain value by use of the INCREMENT option or you may set the number of division spacings by use of the NUMBER option.

LATEST REVISION: October 11, 1984 (Version 9.1)

3.4.28 Xfudge

SUMMARY: Changes the x axis “fudge factor.”

SYNTAX: XFUDGE {ON—OFF—v}

INPUT: {ON} : Turn fudge option on but don’t change fudge factor.

OFF : Turn fudge option off.

v : Turn fudge option on and change fudge factor to v.

DEFAULT VALUES: XFUDGE 0.03

DESCRIPTION: When this option is on, the actual axis limits are changed by a “fudge factor”. **The algorithm for a linearly interpolated axis is:**

$XDIFF = XFUDGE * (XMAX - XMIN)$

$XMIN = XMIN - XDIFF$

$XMAX = XMAX + XDIFF$

where XMIN and XMAX are the data extrema and XFUDGE is the fudge factor. The algorithm is similiar for logarithmically interpolated axes. The fudge option only applies when the axis limits are scaled to the data extrema (see XLIM.)

SEE COMMANDS: XLIM

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.29 Xfull

SUMMARY: Controls plotting of x axis full logarithmic decades.

SYNTAX: XFULL {ON—OFF}

INPUT: {ON} : Turn full decade plotting on.

OFF : Turn full decade plotting off.

DEFAULT VALUES: XFULL ON

DESCRIPTION: Full decade plotting applies only when logarithmic interpolation is being used and when the fixed limits option is off (see XLIM.) When on, the actual axis limits will be set to the first full decade before and after the data limits. When off, the actual data limits will be used.

SEE COMMANDS: XLIM

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.30 Xgrid

SUMMARY: Controls plotting of grid lines in the x direction.

SYNTAX: XGRID {ON—OFF—SOLID—DOTTED}

INPUT: {ON} : Turn x axis grid plotting on but don't change grid type.

OFF : Turn x axis grid plotting off.

SOLID : Turn x axis grid plotting on using solid grid lines.

DOTTED : Turn x axis grid plotting on using dotted grid lines.

DEFAULT VALUES: XGRID OFF

DESCRIPTION: This command controls only x grid lines. The GRID command can be used to control grid lines in both directions.

SEE COMMANDS: GRID

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.31 Xlabel

SUMMARY: Defines the x axis label and attributes.

SYNTAX: XLABEL {ON—OFF—text},{LOCATION location},{SIZE size} **where location is one of the following:**

TOP—BOTTOM—RIGHT—LEFT

and where size is one of the following:

TINY—SMALL—MEDIUM—LARGE

INPUT: {ON} : Turn x axis labeling option on. Don't change text.

OFF : Turn x axis labeling option off.

text : Turn x axis labeling option on. Change text of label. If text contains embedded blanks, it must be enclosed in single quotes.

LOCATION location : Change location of x axis label.

TOP : Top of the plot window.

BOTTOM : Bottom of the plot window.

RIGHT : To the right of the plot window.

LEFT : To the left of the plot window.

SIZE size : Change x axis label text size.

TINY : Tiny text size has 132 characters per line.

SMALL : Small text size has 100 characters per line.

MEDIUM : Medium text size has 80 characters per line.

LARGE : Large text size has 50 characters per line.

DEFAULT VALUES: XLABEL OFF LOCATION BOTTOM SIZE SMALL;

DESCRIPTION: If this option is on, an x axis label is placed on each plot. The size and location of the x axis label can be changed as well as the text of the x axis label itself. The text quality and font used can be set using the GTEXT command.

SEE COMMANDS: GTEXT

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.32 Xlin

SUMMARY: Turns on linear scaling for the x axis.

SYNTAX: XLIN

DEFAULT VALUES: Linear scaling.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.33 Xlog

SUMMARY: Turns on logarithmic scaling for the x axis.

SYNTAX: XLOG

DEFAULT VALUES: Linear scaling.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.34 Xviewport

SUMMARY: Defines the viewport for the x axis.

SYNTAX: XVPORT xmin xmax

INPUT: **xmin** : X axis viewport minimum value. Must be in the range 0.0 to xmax.

xmax : X axis viewport maximum. Must be in the range xmin to 1.0.

DEFAULT VALUES: XVPORT 0.1 0.9

DESCRIPTION: The viewport is the portion of the viewspace (see VSPACE command) in which the actual plot is drawn. The coordinate system used to define the viewspace and viewport is called a virtual coordinate system. A virtual coordinate system does not depend upon the size, shape, or resolution of a particular physical device's display surface. SAC's coordinate system runs from 0.0 to 1.0 in both the x and y directions. The lower left hand corner of the viewspace is the point (0.0, 0.0) and the upper right hand corner of the viewspace is the point (1.0, 1.0). (See the figure on the next page.) The use of this coordinate system lets you position a plot without worrying about a specific output device. The XVPORT and YVPORT commands control where in the viewspace a specific plot is to be drawn. The default values use most of the viewspace for the plot while leaving some room on each side for axes, labels, and a title. You can place a particular plot anywhere you want using these commands. When used in conjunction with the BEGINFRAME and ENDFRAME commands, these commands let you create your own special layout by putting several different plots on the same frame.

EXAMPLES: See the example in the BEGINFRAME documentation.

SEE COMMANDS: VSPACE, BEGINFRAME Principles of Interactive Computer Graphics, Second Edition; William M. Newman and Robert F. Sproull; 1979; McGraw-Hill.

LATEST REVISION: January 8, 1983 (Version 8.0) Viewspace and Viewport Coordinates

3.4.35 Ydiv

SUMMARY: Controls the y axis division spacing.

SYNTAX: YDIV {NICE—INCREMENT v—NUMBER n},{POWER {ON/OFF}}

INPUT: NICE : Use “nice-numbered” division spacings.

INCREMENT v : Set division spacing increment to v.

NUMBER n : Set number of division spacings to n.

POWER {ON} : Turn power option on. When this option is on, SAC may print the division spacings as a number raised to a power of 10.

POWER OFF : Turn power option off.

DEFAULT VALUES: YDIV NICE POWER ON

DESCRIPTION: This command controls the selection of y axis division spacings. Most of the time the default “nice-numbered” spacings are satisfactory. SAC determines these based on the minimum and maximum axis limits, the length of the axis, and the current axis character size. You may also force the division spacing to be a certain value by use of the INCREMENT option or you may set the number of division spacings by use of the NUMBER option.

LATEST REVISION: October 11, 1984 (Version 9.1)

3.4.36 Yfudge

SUMMARY: Changes the y axis “fudge factor.”

SYNTAX: YFUDGE ON—OFF—v

INPUT: ON : Turn fudge option on but don’t change fudge factor.

OFF : Turn fudge option off.

v : Turn fudge option on and change fudge factor to v.

DEFAULT VALUES: YFUDGE 0.03

DESCRIPTION: When this option is on, the actual axis limits are changed by a “fudge factor”. **The algorithm for a linearly scaled axis is:**

$YDIFF = YFUDGE * (YMAX - YMIN)$

$YMIN = YMIN - YDIFF$

$YMAX=YMAX+YDIFF$

where YMIN and YMAX are the data extrema and YFUDGE is the fudge factor. The algorithm is similiar for logarithmically scaled axes. The fudge option only applies when the axis limits are scaled to the data extrema (see YLIM.)

SEE COMMANDS: YLIM

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.37 Yfull

SUMMARY: Controls plotting of y axis full logarithmic decades.

SYNTAX: YFULL {ON—OFF}

INPUT: {ON} : Turn full decade plotting on.

OFF : Turn full decade plotting off.

DEFAULT VALUES: YFULL ON

DESCRIPTION: Full decade plotting applies only when logarithmic scaling is being used and when the fixed limits option is off (see YLIM.) When on, the actual axis limits will be set to the first full decade before and after the data limits. When off, the actual data limits will be used.

SEE COMMANDS: YLIM

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.38 Ygrid

SUMMARY: Controls plotting of grid lines in the y direction.

SYNTAX: YGRID {ON—OFF—SOLID—DOTTED}

INPUT: {ON} : Turn y axis grid plotting on but don't change grid type.

OFF : Turn y axis grid plotting off.

SOLID : Turn y axis grid plotting on using solid grid lines.

DOTTED : Turn y axis grid plotting on using dotted grid lines.

DEFAULT VALUES: YGRID OFF

DESCRIPTION: This command controls only y grid lines. The GRID command can be used to control grid lines in both directions.

SEE COMMANDS: GRID

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.39 Ylabel

SUMMARY: Defines the y axis label and attributes.

SYNTAX: YLABEL {ON—OFF—text},{LOCATION location},{SIZE size} **where location is one of the following:**

TOP—BOTTOM—RIGHT—LEFT

and where size is one of the following:

TINY—SMALL—MEDIUM—LARGE

INPUT: {ON} : Turn y axis labeling option on. Don't change text.

OFF : Turn y axis labeling option off.

text : Turn y axis labeling option on. Change text of label. If text contains embedded blanks, it must be enclosed in single quotes.

LOCATION location : Change location of y axis label.

TOP : Top of the plot window.

BOTTOM : Bottom of the plot window.

RIGHT : To the right of the plot window.

LEFT : To the left of the plot window.

SIZE size : Change y axis label text size.

TINY : Tiny text size has 132 characters per line.

SMALL : Small text size has 100 characters per line.

MEDIUM : Medium text size has 80 characters per line.

LARGE : Large text size has 50 characters per line.

DEFAULT VALUES: YLABEL OFF LOCATION LEFT SIZE SMALL;

DESCRIPTION: If this option is on, a y axis label is placed on each plot. The size and location of the y axis label can be changed as well as the text of the y axis label itself. The text quality and font used can be set using the GTEXT command.

SEE COMMANDS: GTEXT

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.40 Ylin

SUMMARY: Turns on linear scaling for the y axis.

SYNTAX: YLIN

DEFAULT VALUES: Linear scaling.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.41 Ylog

SUMMARY: Turns on logarithmic scaling for the y axis.

SYNTAX: YLOG

DEFAULT VALUES: Linear scaling.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.4.42 Yvport

SUMMARY: Defines the viewport for the y axis.

SYNTAX: YVPORT yvmin yvmax

INPUT: **yvmin** : Y axis viewport minimum value. Must be in the range 0.0 to yvmax.

yvmax : Y axis viewport maximum. Must be in the range yvmin to 1.0.

DEFAULT VALUES: YVPORT 0.15 0.9

DESCRIPTION: The viewport is the portion of the viewspace (see VSPACE command) in which the actual plot is drawn. The coordinate system used to define the viewspace and viewport is called a virtual coordinate system. A virtual coordinate system does not depend upon the size, shape, or resolution of a particular physical device's display surface. SAC's coordinate system runs from 0.0 to 1.0 in both the x and y directions. The lower left hand corner of the viewspace is the point (0.0, 0.0) and the upper right hand corner of the viewspace is the point (1.0, 1.0). (See the figure in the XVPORT documentation.) The use of this coordinate system lets you position a plot without worrying about a specific output device. The XVPORT and YVPORT commands control where in the viewspace a specific plot is to be drawn. The default values use most of the viewspace for the plot while

leaving some room on each side for axes, labels, and a title. You can place a particular plot anywhere you want using these commands. When used in conjunction with the BEGINFRAME and ENDFRAME commands, these commands let you create your own special layout by putting several different plots on the same frame.

EXAMPLES: See the example in the BEGINFRAME documentation.

SEE COMMANDS: VSPACE, XVPORT, BEGINFRAME Principles of Interactive Computer Graphics, Second Edition; William M. Newman and Robert F. Sproull; 1979; McGraw-Hill.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.5 GAM: Graphic Action Module

3.5.1 Fileid

SUMMARY: Controls the file id display found on most SAC plots.

SYNTAX: FILEID {ON—OFF} {TYPE DEFAULT—NAME—LIST
hdrlist}, LOCATION UR—UL—LR—LL}, {FORMAT EQUALS—COLONS—NONAMES}

INPUT: FILEID {ON} : Turn on file id option. Does not change file id type or location.

FILEID OFF : Turn off file id option.

TYPE DEFAULT : Change to the default file id.

TYPE NAME : Use the name of the file as the file id.

TYPE LIST hdrlist : Define a list of header fields to display in the fileid.

LOCATION UR : Place file id in upper right hand corner.

LOCATION UL : Place file id in upper left hand corner.

LOCATION LR : Place file id in lower right hand corner.

LOCATION LL : Place file id in lower left hand corner.

FORMAT EQUALS : Format consists of header field name, an equals sign, and the header field value.

FORMAT COLON : Format consists of header field name, a colon, and the value.

FORMAT NONAMES : Format consists of header field value only.

DEFAULT VALUES: FILEID ON TYPE DEFAULT LOCATION UR
FORMAT NONAMES

DESCRIPTION: This command controls the file id that is displayed on most SAC plot formats. The file id identifies the content of the plot. The default file id consists of the event name, the station name and component, and the zero date and time. The name of the file can be substituted for the default id if desired. A special file id can be defined and displayed. This special file id can consist of up to 10 SAC header fields. The location and format of the fileid can also be changed.

EXAMPLES: To put the filename in the upper left corner:

u: FILEID LOCATION UL TYPE NAME

To define a special file id consisting of the station component, latitude, and longitude:

u: FILEID TYPE LIST KSTCMP STLA STLO

To include the name of the header field followed by a colon:

u: FILEID FORMAT COLON

LATEST REVISION: October 11, 1984 (Version 9.1)

3.5.2 Filenumber

SUMMARY: Controls the file number display found on most SAC plots.

SYNTAX: FILENUMBER {ON—OFF}

INPUT: FILENUMBER ON : Turn on file number option.

FILENUMBER {OFF} : Turn off file number option.

DEFAULT VALUES: FILENUMBER OFF

DESCRIPTION: This command controls the file number that is displayed on most SAC plots. When filenumber is on, the file number appears on the plot. This can be used to identify a specific waveform by number when a command requires the information.

EXAMPLES:

LATEST REVISION: February 5, 1997 (Version 53)

3.5.3 Picks

SUMMARY: Controls the display of time picks on most SAC plots.

SYNTAX: PICKS {ON—OFF} {pick type},{WIDTH v},{HEIGHT v}

where pick is one of the following

O—A—F—T_n, n=0...9

where type is one of the following

VERTICAL—HORIZONTAL—CROSS

INPUT: PICKS ON : Turn on pick display.

PICKS OFF : Turn off pick display.

pick : The name of a SAC time pick header variable: ,BREAK
O—A—F—T_n, n=0...9

VERTICAL : A vertical line at time pick. Pick id at bottom right of line.

HORIZONTAL : A horizontal line at the data point nearest the time pick. Pick id is placed above or below the line.

CROSS : A vertical line at the time pick and a horizontal line at the nearest data point.

WIDTH v : Change the width of the pick display to v.

HEIGHT v : Change the height of the pick display to v. Height and width refer to HORIZONTAL and CROSS only and are in fractions of full plot size.

DEFAULT VALUES: PICK ON WIDTH 0.1 HEIGHT 0.1; all pick types are VERTICAL.

DESCRIPTION: This command controls the display of time pick information on most SAC plots. These time picks identify previously defined times of interest such as phase arrivals, event origin, etc. When this display is on, each defined time pick is displayed on the plot at the time of the pick with a time pick id near the line. The time pick id is a header variable 8 characters in length. KA, KF, KO, and KT_n are the time pick ids for A, F, O, and T_n respectively. If the time pick id is not defined, the name of the pick itself is used. Each pick may be displayed as a vertical line, a horizontal line, or a cross.

EXAMPLES: To display time picks T4, T5, and T6 as crosses and to change the height and **width of the crosses:**

u: PICKS T4 C T5 C T6 C W 0.3 H 0.1

Other time pick display types will remain unchanged.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.5.4 Plot

SUMMARY: Generates a single-trace single-window plot.

SYNTAX: PLOT {PRINT {pname} }

INPUT: PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DESCRIPTION: Each data file is displayed in a separate plot. The total size of the plot is determined by the current viewport (see XVPORT and VPORT.) The y axis limits for each plot can be scaled to the data file's extrema or they can have fixed limits. See the YLIM command for details. The x axis limits are controlled by the XLIM command. A user controllable file identification (see FILEID) is generated for each file in the plot. Time picks can be displayed (see PICKS). If you are plotting to a graphics terminal or workstation, SAC will pause between each plot to give you an opportunity to examine the plot. It will type "Waiting" in the text area and wait for your response. You can type a carriage-return to see the next plot, the keyword "go" (or "g") to plot the remainder of the files without pausing, or the keyword "kill" (or "k") to not plot the remainder of the files at all.

ERROR MESSAGES: 1301: No data files read in.

SEE COMMANDS: XVPORT, YVPORT, XLIM, YLIM, FILEID, PICKS, FILENUMBER

LATEST REVISION: January 8, 1983 (Version 8.0)

3.5.5 Plot1

SUMMARY: Generates a multi-trace multi-window plot.

SYNTAX: PLOT1 {ABSOLUTE—RELATIVE},{PERPLOT {ON—OFF—n}}
{PRINT {pname} }

INPUT: ABSOLUTE : Plot files treating time as an absolute. Files with different begin times will be shifted relative to each other.

RELATIVE : Plot all files relative to their begin time.

PERPLOT {ON} : Plot n files per frame. Use old value for n.

PERPLOT OFF : Plot all files on one frame.

PERPLOT n : Plot n files per frame.

PRINT {pname} : Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

ALTERNATE FORMS: PERPLOT ALL has the same meaning as PERPLOT OFF.

DEFAULT VALUES: PLOT1 ABSOLUTE PERPLOT OFF

DESCRIPTION: Each data file shares a common axis in the x direction, but each has a separate subplot region in the y direction. The total size of the plot is determined by the current viewport (see XVPORT and YVPORT.) The size of each subplot is determined by this viewport and the number of files plotted on each frame. The y axis limits for each subplot can be scaled to that data file's extrema or they can have fixed limits. See the YLIM command for details. The x axis limits can also be fixed (see XLIM) or scaled to the data. **There are two types of x axis scaling for this type of plot:** relative and absolute. In absolute scaling the x axis limits become the smallest minimum and the largest maximum for the active memory files. Time differences measured between points on different subplots will be correct. In relative scaling mode, the x axis will run from zero to the maximum time differential (i.e., the maximum difference between end time and begin time) for the active memory files. Each file will be plotted from the left edge of the plot, corresponding to zero on the x axis. The actual value corresponding to this zero for each file will be given below the name of the file. This type of scaling is useful if you are cutting the files relative to some time pick, say the first arrival time. It is then easy to see the similarities or differences between the wave forms of each file. An

example of this type of plot is given on a following page. A user controllable file identification (see FILEID) is generated for each file in the plot. Time picks can be displayed (see PICKS).

EXAMPLES: The following example is from an earthquake in the western U.S. recorded at the four LLNL DSS stations in Elko, Kanab, Landers, and Mina. The zero time (KZDATE and KZTIME) has been set to the event origin time. The commands used **to generate the plot on the next page are given below:**

```
u: CUT -5 200
u: READ *V
s: ELK.V KNB.V LAC.V MNV.V
u: FILEID LOCATION UL TYPE LIST KSTCMP
u: TITLE 'Regional earthquake: &1,KZTIME& &1,KZDATE&'
u: QDP 2000
u: PLOT1
```

Note the use of a UNIX wildcard character in the READ command, the echoing of the filelist by SAC, the specification of a special file id, and the evaluation of several header variables to create the title.

ERROR MESSAGES: 1301: No data files read in.

SEE COMMANDS: XLIM, YLIM, FILEID, PICKS, FILENUMBER

LATEST REVISION: January 8, 1983 (Version 8.0) Sample PLOT1
Output

3.5.6 Plot2

SUMMARY: Generates a multi-trace single-window (overlay) plot.

SYNTAX: PLOT2 {ABSOLUTE—RELATIVE} {PRINT {pname} }

INPUT: ABSOLUTE : Plot files treating time as an absolute. Files with different begin times will be shifted relative to the first file.

RELATIVE : Plot each file relative to it's own begin time.

PRINT {pname} : Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES: PLOT2 ABSOLUTE

DESCRIPTION: All files in the data file list are plotted in the same plot window. An optional legend containing the plot symbol and file name can be generated. Fixed x and y axis limits may be defined before using this command. See the XLIM and YLIM commands. The plot is sized to the extrema of the entire file list if fixed limits are not requested. An example is given on the next page. The location of the legend is controlled by the FILEID command.

Unlike PLOT and PLOT1, PLOT2 will plot spectral data. Real/Imaginary data is plotted as Real vs. Frequency. Amplitude/Phase data is plotted as Amplitude vx. Frequency. Imaginary and Phase information is ignored. Spectral data is always plotted in relative mode.

Note: If for some reason, the user has both time-series data and spectral data in memory at the same time and does not elect to use the RELATIVE option, the time-series files will be plotted ABSOLUTE and the spectral files will be plotted RELATIVE. Relative mode means relative to the first file. So the order of the files in memory will effect the relation of the plots with respect to each other.

EXAMPLES: The commands used to generate the example plot are given below:

```
u: READ MNV.Z.AM KNB.Z.AM ELK.Z.AM
u: XLIM 0.04 0.16
u: YLIM 0.0001 0.006
u: LINLOG
u: SYMBOL 2 INCREMENT
u: TITLE 'Rayleigh Wave Amplitude Spectra for NESSEL'
u: XLABEL 'Frequency (Hz)'
u: PLOT2
```

ERROR MESSAGES: 1301: No data files read in.

SEE COMMANDS: XLIM, YLIM, FILEID, FILENUMBER

LATEST REVISION: June 22, 1999 (Version 0.58) Sample PLOT2

Output

3.5.7 Plotalpha

SUMMARY: Reads alphanumeric data files on disk into memory and plots the data to the current output device. This differs from readalpha followed by plot because it allows you to plot a label with each data point.

SYNTAX: READALPHA {options} {filelist} **where options is one or more of the following:**

MORE DIR CURRENT—name FREE—FORMAT text CONTENT text
PRINT {printer}

ALL options MUST precede any element in the filelist. The last two options may also be placed on the first line of file itself.

INPUT: MORE : Append the new data files after the old ones in memory. If this option is missing, the new data replaces the old data in memory. See the READ command for more details about this option.

DIR CURRENT : Read and plot all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name : Read and plot all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

FREE : Read and plot the data in the filelist in free format (space delimited) mode.

FORMAT text : Read and plot the data in the filelist in fixed format mode. The format statement to use is given in text.

CONTENT text : Define the content of the data in the filelist. The meaning of the content text is described below.

PRINT {pname} : Print the resultant plot. If pname is specified, print to named printer, else use default printer.

filelist : A list of alphanumeric files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DEFAULT VALUES: PLOTALPHA FREE CONTENT Y. DIR CURRENT

DESCRIPTION: All commands in SAC work on the data that is currently in memory. This data in memory is analogous to the temporary or working files used by a text editor. The READ command reads binary SAC data files into memory. This command can be used to read a wide variety of alphanumeric data files into memory. These files can be in a fixed format or in free format. They may contain evenly or unevenly spaced data. They may contain more than one set of data. There may be only one label and the label is not retained in memory with the data. The simplest use of this command is free field input of a Y data set. This is also the default. Free field input of X-Y pairs can be done by simply changing the content option. By combining the fixed format and content options, this command can also be used to read very complicated formatted output from other programs directly into SAC. Multiple Y data sets can be read from the same file using this method. Only a single X data set is allowed. The basic header variables needed for processing are computed. These are NPTS, B, E, DELTA, LEVEN, DEPMIN, DEPMAX, and DEPMIN. If there is only a single Y data set, the name of the data file in memory will be the same as that of the alphanumeric disk file. If there are multiple Y data sets in the file, a two digit sequence number is appended to the file name. Each line of the alphanumeric data file is read in either free format or using the format statement provided. Each line can be up to 160 characters long. In the case of a free format file, the number of data entries in each line is also determined. The content field is then used to determine what to do with each of these data entries. Each specific character in the context field represents a different kind of data element and the order of these characters mimics the order of the data in each line of the file. **The meanings of the allowed characters in the content field are given below:**

L = Next entry is the label to plot with each data point (only one per data set). Y = Next entry belongs to Y (dependent variable) data set.

X = Next entry belongs to X (independent variable) data set.

N = Next entry belongs to next Y data set.

P = Next pair of entries belong to X and Y data sets.

R = Next pair of entries belong to Y and X data sets.

I = Ignore (skip) this data element.

An optional repetition count may follow any of the above characters. This repetition count is a one or two digit integer and has the same meaning as repeating the content character that number of times. A period (“.”) is an infinite repetition count and means use the last characters meaning to decode the remaining data elements in the line. The period can only appear at the end of a content field.

EXAMPLES: To read and plot X-Y pairs in free format where the first field is the label:

u: PLOTALPHA CONTENT LP FILEA

You can’t break an X-Y pair between lines in the file.

ERROR MESSAGES: 1301: No data files read in.

- haven’t given a list of files to read. - none of the files in the list could be read. **1020: Invalid inline function name:**

- Expected inline function. Preceed parenthesis with an atsign. **1320:** Available memory too small to read file

1314: Data file list can’t begin with a number.

1315: Maximum number of files in data file list is

WARNING MESSAGES: 0101: opening file

0108: File does not exist:

HEADER CHANGES: B, E, DELTA, LEVEN, DEPMIN, DEPMAX, DEPMEN.

SEE COMMANDS: READ, WRITE, READALPHA

LATEST REVISION: July 22, 1992 (Version 10.6f)

3.5.8 Plotc

SUMMARY: Annotates SAC plots and creates figures using cursor.

SYNTAX: PLOTc {REPLAY—CREATE} {FILE—MACRO filename}, {BORDER {ON—OFF}}

INPUT: REPLAY : Replay or replot an existing file or macro. The difference between a file and a macro is described below.

CREATE : Create a new file or macro.

FILE {filename} : Replay or create a file. The previous file is used if filename is omitted.

MACRO {filename} : Replay or create a macro.

BORDER {ON} : Turn border around plot on.

BORDER OFF : Turn border around plot off.

DEFAULT VALUES: PLOT CREATE FILE OUT BORDER ON

DESCRIPTION: This command lets you annotate SAC plots and create figures for meetings and reports. A device with cursor capability is required. You “build” a figure by placing objects and text on the terminal screen. The cursor position determines where an object will be drawn and the character typed determines what object is to be drawn. Objects include circles, rectangles, n-sided polygons, lines, arrows, and arcs. Several ways of placing text are included. This command creates two different type of output files, simple files and macro files. Both are alphanumeric files that can be changed using an editor. They contain the history of the cursor responses and locations from a single execution of the PLOT command. A macro file, once created, can be used in more than one plot or figure. It can be scaled in size and can also be rotated. A simple PLOT filename is the name you request with a “.PCF” appended to it. A macro file has a “.PCM” appended to its’ name. This provides a check for SAC when you request a particular file and also lets you distinguish these files in your directories. When you create a new file or macro, SAC draws a rectangle on the screen showing you the allowable area for the figure. It then turns the cursor on in the middle of this area. You move the cursor to the desired location and type a character representing the object you want drawn or the action you want to occur. There are two types of cursor options, action and parameter-setting. The action options do something (draw a polygon, place text, etc.) How they do that action is based in part upon the current values of the parameter-setting options (how many sides on the polygon, what size text to draw, etc.) This distinction is similiar to the idea of action and parameter-setting commands in SAC itself. The tables on the following pages list the action and parameter-setting options. When you replay a file or macro, the figure is redrawn on the terminal screen and then the cursor

is turned on. You may then add to the file or macro as if you were creating it for the first time. When you have created a figure that you want to send to a different graphics device, use the BEGINDEVICES command to temporarily turn off the terminal and turn on the other device. Then simply replay the file. To annotate a SAC plot, execute the VSPACE command to set up the correct aspect ratio (see below), execute the BEGINFRAME command to turn off automatic framing, execute the desired SAC plot command, execute the PLOT command (in create or replay mode), and then execute the ENDFRAME command to resume automatic framing.

EXAMPLES: An example of the use of PLOT to add annotation to a standard SAC plot is the figure in the BANDPASS command description of this manual. The commands used to create that figure are given below with comments given in **parentheses**:

```

u: FG IMPULSE NPTS 1024 \ \ (generate filter response)
u: LOWPASS C2 NPOLES 7 CORNER 0.2 TRANBW 0.25 A 10
u: FFT
u: AXES ONLY LEFT BOTTOM \ \ (set up desired plot options)
u: TICKS ONLY LEFT BOTTOM
u: BORDER OFF
u: FILEID OFF
u: QDP OFF
u: VSPACE 0.75 \ \ (see discussion below)
u: BEGINFRAME \ \ (turn off automatic framing)
u: PLOTSP AM LINLIN \ \ (plot filter response)
u: PLOT CREATE FILE BANDPASS \ \ (create annotation)
u: ENDFRAME \ \ (turn automatic framing back on)

```

PLOTSP was used to produce the curve of the filter response and the two axes. PLOT was used interactively to produce the annotation (i.e., the lines, arrows, and labels.) The viewspace command constrains the plot be the largest **enclosed area of the graphics screen that has an \ \ (y:x) aspect ratio of 3:4**. This is necessary so that when the output is later sent to the SGF device **which also has a 3:4 aspect ratio**, everything will be plotted correctly. At this point you would have a file called

“BANDPASS.PCF” containing the annotations for this plot. To write this annotated plot to the SAC **graphics file**:

u: BEGINDEVICES SGF \ \ (select the SGF device)

u: BEGINFRAME \ \ (turn off automatic framing)

u: PLOTSP \ \ (plot filter response again)

u: PLOTTC REPLAY \ \ (replay the annotation)

u: ENDFRAME \ \ (turn automatic framing back on)

A SAC graphics file will be created containing the annotated plot. Two examples (one somewhat frivolous) of the use of PLOTTC to create figures and viewgraphs are given on the following pages. The replay files are also shown. (It is an exercise left to the reader to determine which of the examples is frivolous.)

(1) The circle and sector opcodes only produce correct output when you have set the viewspace to a square one (VSPACE 1.0). Otherwise, they produce an ellipse with the ratio of the minor to major axis equal to the aspect ratio of the viewspace.

(2) All all of the opcodes except text are scaled to fit in the graphics window. The text sizes aren’t currently scaled. This creates a problem when you create a figure and want to enclose some text in a rectangle or a circle. In this case the graphics window must be the same size as the output page in order to avoid misalignment. This can be achieved by using the WINDOW command to set the horizontal (x) size of the window to be 0.75 and the vertical (y) size to be 0.69. **For example:** WINDOW 1 X 0.05 0.80 Y 0.05 0.74 This command must be executed before the window is created (i.e. before the BEGINWINDOW or BEGINDEVICES command.)

(3) The text feature of this command works only in SunView graphics windows.

SEE COMMANDS: VSPACE, BEGINDEVICES, BEGINFRAME, ENDFRAME

LATEST REVISION: March 20, 1992 (Version 10.6e) Sample PLOTTC Figure Sample PLOTTC Figure

TABLE: Action Options

char meaning A Draw an arrow from ORIGIN to CURSOR. B Draw

border tick marks around plot region. C Draw a circle centered at ORIGIN through CURSOR. D Delete last action option from replay file. G Set ORIGIN and make it global. L Draw a line from ORIGIN to CURSOR. M Invoke a macro at CURSOR. Enter name of macro, scale factor, and rotation angle. Previous values are used if omitted. Defaults are OUT, 1., and 0. O Set ORIGIN at CURSOR. N Draw an n-sided polygon centered at ORIGIN with one vertex at CURSOR. Q Quit PLOT. R Draw a rectangle with opposing corners at ORIGIN and CURSOR. S Draw a sector of a circle centered at ORIGIN through CURSOR. Move CURSOR to define the sector angle. Type an S to get the sector whose angle is less than 180 degrees or C to get its' complement. T Place a single line of text at cursor. Text is ended by a carriage-return. U Place multiple lines of text at cursor. Text is ended by a blank line.

Notes:

- CURSOR is the current cursor location
- ORIGIN is normally the last cursor location
- The G option forces ORIGIN to remain fixed
- The O option allows ORIGIN to move again
- The Q option is not automatically copied to the file but may be added to it with a text editor. If SAC does not see a Q in the file during replay mode, it goes back into cursor mode after displaying the contents of the file. This lets you append more options to the end of a file. If SAC does see a Q in the file, it displays the contents and ends PLOT.
- A line beginning with an asterisk is treated as a comment line.

3.5.9 Plotdy

SUMMARY: Creates a plot with error bars.

SYNTAX: PLOTDY {ASPECT ON—OFF} {PRINT pname} name—number
name—number { name—number }

INPUT: **ASPECT** : ON maintains a 3/4 aspect ratio. OFF allows the aspect ratio to vary with the dimensions of the window.

PRINT {pname} : Print the resultant plot. If a printer name is

given, print to that printer, else use default printer.

name : The name of a data file in the data file list.

number : The number of a data file in the data file list.

DESCRIPTION: This command allows you to plot a data set with error bars. The first data file you select (either by name or number) is plotted along the y axis. The second data file is the dy value. If a third data file is selected it is the positive dy value. Assume you have an evenly spaced ascii file that contains two columns of numbers. The first is the y-value. The second column is the dy-value. You wish to read these into SAC and plot the data with error bars.

u: READALPHA CONTENT YY MYFILE

u: PLOTDY 1 2

1301: No data files read in.

LATEST REVISION: July 22, 1992 (Version 10.6f)

3.5.10 Plotpk

SUMMARY: Produces a plot for the picking of arrival times.

SYNTAX: PLOTPK options **where options are one or more of the following:**

{PERPLOT {ON—OFF—n}}, {BELL {ON—OFF}}, {ABSOLUTE—RELATIVE},
{REFERENCE {ON—OFF—v}}, {MARKALL {ON—OFF}}, {SAVELOCS
{ON—OFF}}

INPUT: PERPLOT {ON} : Plot n files per frame. Use old value for n.

PERPLOT OFF : Plot all files on one frame.

PERPLOT n : Plot n files per frame.

BELL {ON} : Ring bell on each keystroke in active window.

BELL OFF : keystrokes are silent.

ABSOLUTE : Display times in absolute (GMT) format.

RELATIVE : Display times relative to each file's zero time.

REFERENCE {ON} : Turn reference line display on.

REFERENCE OFF : Turn reference line display off.

REFERENCE v : Turn reference line display on and change reference value to v.

MARKALL {ON} : Store header picks in all of the files displayed on a particular plot.

MARKALL OFF : Store header pick only in the file marked by the horizontal cursor.

SAVELOCS {ON} : Save pick locations (from L cursor command) in the blackboard.

SAVELOCS OFF : Do not save pick locations in the blackboard.

ALTERNATE FORMS: GMT may be used instead of ABSOLUTE and ZERO may be used instead of RELATIVE.

DEFAULT VALUES: PLOTPK PERPLOT OFF ABSOLUTE REFERENCE OFF MARKALL OFF SAVELOCS OFF

DESCRIPTION: The format of the PLOTPK plot is similiar to the PLOT1 plot. This plot requires a terminal with a cursor. When the cursor comes on and the bell rings, you enter single character responses to perform the various functions. Some but not all responses produce graphic output on the screen. Error and output messages are printed at the top of the screen. Picks that are currently in the header are automatically displayed on the screen. Output from the various cursor responses can be directed to the SAC header, to a alphanumeric pick file if open (see OAPF), to a HYPO pick file if open, and to the terminal. If you use the SAVELOCS option to save the cursor locations from the L cursor option in the blackboard, the following blackboard variables will be **defined:**

NLOCS: The number of locations picked during the execution of this command. This is initialized to 0 each time PLOTPK is invoked and incremented by 1 each time a cursor location pick is made.

XLOCn: The x value for the nth cursor location pick. This will be the GMT time of the pick if the reference time fields in the header are defined. Otherwise, this will be an offset time.

YLOCn: The y value for the nth cursor location pick.

The table on the following page gives the valid cursor options for this plot.

HEADER CHANGES: Depending upon user responses any of A, KA, F, KF, Tn, KTn.

ERROR MESSAGES: 1301: No data files read in.

1202: Maximum number of vars sections exceeded:

WARNING MESSAGES: 1502: Bad cursor position. Please retry.

- cursor is positioned outside of the plot window. - occasionally the Tektronix sends the wrong position to SAC. Move the cursor a little and retry. **1503:** Invalid character. Please retry.

- A character was input that SAC didn't recognize as a legal response.

1905: Need an integer. Retry.

- Didn't input an integer following the T response. **1906:** Need an integer in the range 0 to 4. Retry.

- Didn't input a 0, 1, 2 or 3 after a Q response. **1907:** HYPO line already written.

1908: HYPO pick file not open.

1909: Can't compute waveform.

- Adjust cursor position and retry. Plot is always in ABSOLUTE mode.

SEE COMMANDS: PLOT1, OHPF, OAPF, APK

LATEST REVISION: March 13, 1997 (Version 00.53a)

3.5.11 Plotpm

SUMMARY: Generates a "particle-motion" plot of pairs of data files.

SYNTAX: PLOT1 {PRINT {pname} }

INPUT: PRINT {pname} : Print the resultant plot. If a printer name is specified, print to that printer, else use default printer.

DESCRIPTION: In a particle-motion plot one evenly spaced file is plotted against another. For each value of the independent variable, normally time, the value of the dependent variable of the first file is plotted along the y axis and the value of the dependent variable of the second file is plotted along the x axis. For a pair of seismograms this type of plot shows the motion of a "particle" in the plane of the two seismograms as a function of time. A square plot is generated, with the limits along each axis being

the minimum and maximum values of the dependent variable. Annotated axes are generated along the bottom and left. Axes labels and title can be set by the XAXIS, YAXIS, and TITLE commands. If no x and y axis labels are set, then the name and azimuth of the station are used as axes labels. The XLIM command can be used to control how much of each file to plot.

EXAMPLES: To create a particle-motion plot of two seismograms, XYZ.T and XYZ.R and set **up your own axes labels and title:**

```
u: READ XYZ.T XYZ.R
u: XLABEL 'Radial component'
u: YLABEL 'Transverse component'
u: TITLE 'Particle-motion plot for station XYZ'
u: PLOTPM
```

If you wanted to plot only a small part of each file around the first arrival time, you could use the **XLIM** command:

```
u: XLIM A -0.2 2.0
u: PLOTPM
```

You could also use PLOTPK, possibly in a different graphics window as in this example, to mark which portion of the files you wanted to see in the particle **motion plot:**

```
u: BEGINWINDOW 2
u: PLOTPK
u: ... mark the portion you want using X and S
u: ... terminate PLOTPK with a Q
u: BEGINWINDOW 1
u: PLOTPM
```

LATEST REVISION: May 15, 1987 (Version 10.2)

3.5.12 Plotxy

SUMMARY: Plots one or more data files versus another data file.

SYNTAX: PLOTXY name—number name—number { name—number
... }

INPUT: **name :** The name of a data file in the data file list.

number : The number of a data file in the data file list.

DESCRIPTION: This command lets you plot one or more data files versus another data file. The first data file you select (either by name or number) becomes the independent variable and is plotted along the x axis. The remainder of the data files you select become the dependent variables and are plotted along the y axis. All of the graphics environment commands such as TITLE, LINE, and SYMBOL can be used to control attributes about the plot. This command can be used to easily plot multi-columned data that has been read in with the READALPHA command. In this case it can be viewed as a spreadsheet like plotting command. An example is given below.

EXAMPLES: Assume you have an ascii file that contains four columns of numbers. You wish to read these into SAC and plot various columns versus each other. The following commands would read this file in and store it as four separate data files inside SAC, turn linestyle incrementing on and then plot the first, **third, and fourth columns versus the second column:**

u: READALPHA CONTENT YNNN MYFILE

u: LINE INCREMENT ON

u: PLOTXY 2 1 3 4

LATEST REVISION: April 21, 1989 (Version 10.4c)

3.5.13 Print

SUMMARY: Prints the most recent SGF file. This command requires that at least one SGF file has been produced.

SYNTAX: PRINT {printer}

INPUT: printer : sends output to the named printer, if no printer name is supplied, it will print to the system's default printer.

Note: PRINT will not work if the SGF device has remained off since boot. Use BEGINDEVICES to turn on the SGF device. Use the SGF command to set preferred behavior for the SGF device. The SGF command has an overwrite option which prevents disk file buildup by clobbering previous SGF files with the new ones.

DEFAULT VALUES: PRINT

ERROR MESSAGES: 2405: Cannot PRINT: no SGF files produced.

SEE COMMANDS: SGF, BEGINDEVICES

LATEST REVISION: April 22, 1999 (Version 0.58)

3.5.14 Setdevice

SUMMARY: Defines a default graphics device to use in subsequent plots.

SYNTAX: SETDEVICE name

INPUT: name : The name of a graphics device.

DESCRIPTION: This command lets you define the name of a default graphics device to use in subsequent plots. This command is only useful before you do any plotting. It should be placed in your default macro file. You can override the name specified in this command by using the BEGINDEVICES command. See the section on Graphics Devices in the SAC Users Manual. Also see the section on Macros for information on specifying and using a default macro file.

SEE COMMANDS: BEGINDEVICES

LATEST REVISION: May 15, 1987 (Version 10.2)

3.5.15 Xlim

SUMMARY: Determines the plot limits for the x axis.

SYNTAX: XLIM {ON—OFF—pdw—SIGNAL}

INPUT: {ON} : Turn x limits on but don't change limits.

OFF : Turn x limits off.

pdw : Turn x limits on and set limits to a new "partial data window." A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to plot. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

SIGNAL : Equivalent to typing: A -1 F +1.

DEFAULT VALUES: XLIM OFF

DESCRIPTION: When this option is on, fixed plot limits are used for the x axis. When this option is off, the limits are scaled to the data. Fixed x limits can be used to “blowup” part of the data currently in memory.

EXAMPLES: In these examples we assume time is the independent variable and seconds are the units.

B 0 30: First 30 secs of the file.

A -10 30: From 10 secs before to 30 secs after first arrival.

T3 -1 T7: From 1 sec before T3 time pick to T7 time pick.

B N 2048: First 2048 points of file.

30.2 48: 30.2 to 48 secs relative to file zero.

SEE COMMANDS: CUT

LATEST REVISION: January 8, 1983 (Version 8.0)

3.5.16 Ylim

SUMMARY: Determines the plot limits for the y axis.

SYNTAX: YLIM {ON—OFF—ALL—min max—PM v}

INPUT: {ON} : Turn y limits option on, but don’t change limits.

OFF : Turn y limits option off.

ALL : Scale y limits to the minimum and maximum of all files in memory.

min max : Turn fixed y option on and change limits to min and max.

PM v : Turn fixed y option on and change limits to minus and plus the absolute value of v. ,SKIP You may define different y limit options for each file in memory if you wish. The first entry in the command applies to the first file in memory, the second entry to the second file, etc. The last entry applies to the remainder of the files in memory.

DEFAULT VALUES: YLIM OFF

DESCRIPTION: When this option is on, fixed limits are used in plotting. When off, the limits are scaled to the data. The limits can also be scaled to the entire data set if desired. Different values may be set for each file in memory.

EXAMPLES: Consider the following set of commands:

u: YLIM 0.0 30.0 ALL OFF

u: READ FILE1 FILE2 FILE3

u: PLOT

FILE1 would be plotted with y limits of 0.0 and 30. FILE2 would be scaled to the minimum and maximum values of all files in memory. FILE3 would be scaled to its own minimum and maximum values. If more than three files were read in, they would also be scaled to their own minimum and maximum values.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.6 SAM: Spectral Analysis Module

3.6.1 Bandpass

SUMMARY: Applies an IIR bandpass filter.

SYNTAX: BANDPASS {BUTTER—BESSEL—C1—C2},{CORNERS
v1 v2},

{NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}

INPUT: BUTTER : Apply a Butterworth filter.

BESSEL : Apply a Bessel filter.

C1 : Apply a Chebyshev Type I filter.

C2 : Apply a Chebyshev Type II filter.

CORNERS v1 v2 : Set corner frequencies to V1 and V2.

NPOLES n : Set number of poles to N {range: 1-10}.

PASSES n : Set number of passes to N {range: 1-2}.

TRANBW v : Set the Chebyshev transition bandwidth to v.

ATTEN v : Set the Chebyshev attenuation factor to v.

DEFAULT VALUES: BANDPASS BUTTER CORNER 0.1 0.4 NPOLES
2 PASSES 1 TRANBW 0.3 ATTEN 30.

DESCRIPTION: A set of Infinite Impulse Response (IIR) filters is available in SAC. **These recursive digital filters are all based upon classical analog designs:** Butterworth, Bessel, Chebyshev type I, and

Chebyshev type II. These analog prototype filters are mapped to digital filters via the bilinear transformation, a transformation which preserves the stability of the analog prototypes. A complete description of this method of design can be found in the reference given below. However, it is not necessary to read that description, unless you want complete control over the more complicated Chebyshev filters. Generally speaking, the Butterworth filter is a good choice for most applications, since it has a fairly sharp transition from pass band to stop band, and its group delay response is moderate. The Butterworth filter is the default filter type. Its 3 db point is at the designated cutoff frequency. The Bessel filter is best for those applications which require linear phase without twopass filtering. Its amplitude response is not very good. The SAC Bessel filters have been normalized so that their 3 db points are also at the designated cutoff frequency. The two Chebyshev filters are included for situations which require very rapid transitions from pass band to stop band. Although they have good magnitude discrimination, their group delay responses are the worst among the filters contained in SAC. Some caution must be exercised in applying these filters. First, all recursive filters have non-linear phase, which can result in some dispersion of filtered waveforms. For applications where the phase of the resulting filtered waveform is important, a zero-phase implementation of the recursive filters is provided. Zero-phase filtering is possible by running the filter forward and backward over the data, instead of just forward over the data. This two-pass operation results in a effective filter magnitude response which is the square of the original magnitude response. It also results in a non-causal filter impulse response, which can leave a signal containing a sharp time onset with a ringing precursor. For this reason, you should not measure arrival times of data that has been filtered using this two-pass option. For cases where signal precursors cannot be tolerated, such as onset picking operations, it may not be a good idea to do two-pass filtering. Second, the filters can become numerically unstable if the width of the filter pass band is very small compared to the folding frequency of the data. The problem is only aggravated by increasing the number of poles in the filter. Situations that seem to require an exceptionally narrow band

filter can be handled more reliably by decimation, filtering with a filter of more moderate band width, and interpolation to the original sampling rate. Recourse to this resampling strategy should be made when the filter band width drops below a few percent of the folding frequency. Generally, the filter will have a sharper transition from pass band to stop band as the number of poles is increased. However, there are penalties for using a large numbers of poles. Filter group delays generally get wider as the number of poles increases, resulting in worse dispersion of the filtered waveform. Applications that appear to require more than three or four poles should probably be reconsidered. The design of Butterworth and Bessel filters is particularly simple. You simply specify the cutoffs of the filter and the number of poles. Chebyshev filters are more complicated to design. In addition to cutoffs and number of poles, you must supply a transition band width, and a stop band attenuation factor for the analog prototype filter. The transition band width is the width of the region between the filter pass band and stop band. It is specified as a fraction of the analog prototype pass band width. Due to the non-linear warping of the frequency axis of the bilinear transformation, the transition band width of the recursive digital filter may be smaller than that specified in the design. In SAC, the analog prototype filter cutoffs are compensated to ensure that they map to the requested cutoffs after the bilinear transformation is performed. The same is not true of the stop band edges. Consequently, if precisely located stop band edges are necessary, you must compensate for this shrinkage when choosing your cutoffs. The stop band attenuation is specified as the ratio of the pass band gain to the stop band gain. The accompanying figure illustrates the use of the Chebyshev type II filter design parameters, and the frequency response that result. (A lowpass filter is used but the figure applies equally as well to the other filters.) The number of poles is 7, the stop band attenuation factor is 10 (20 db), the cutoff 0.2 Hz, and the transition band width ratio is 0.25. The Chebyshev type I filter has ripples in the pass band and a monotonic stop band whereas the type II filter has ripples in the stop band and is monotonic in the pass band.

EXAMPLES: To apply a four-pole Butterworth with corners at 2 and

5 Hz.

u: BANDPASS NPOLES 4 CORNER 2 5

To later apply a two-pole two-pass Bessel with the same corners.

u: BP N 2 BE P 2

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

1002: Bad value for

- corner frequency larger than Nyquist frequency.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 8, 1983 (Version 8.0) Magnitude Frequency Response of Chebyshev Type II Filter

3.6.2 Bandrej

SUMMARY: Applies an IIR bandreject filter.

SYNTAX: BANDREJ {BUTTER—BESSEL—C1—C2},{CORNERS v1 v2}, {NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}

INPUT: BUTTER : Apply a Butterworth filter.

BESSEL : Apply a Bessel Filter.

C1 : Apply a Chebyshev Type I filter.

C2 : Apply a Chebyshev Type II filter.

CORNERS v1 v2 : Set corner frequencies to v1 and v2.

NPOLES n : Set number of poles to n {range: 1-10}.

PASSES n : Set number of passes to N {range: 1-2}.

TRANBW v : Set the Chebyshev transition bandwidth to V.

ATTEN v : Set the Chebyshev attenuation factor to V.

DEFAULT VALUES: BANDREJ BUTTER CORNER 0.1 0.4 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.

DESCRIPTION: See the BANDPASS command for definitions of the filter parameters and descriptions on how to use them.

EXAMPLES: To apply a four-pole Butterworth with corners at 2 and 5 Hz.

u: BANDREJ NPOLES 4 CORNER 2 5

To apply a two-pole two-pass Bessel with the same corners.

u: BR N 2 BE P 2

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

1002: Bad value for

- corner frequency larger than Nyquist frequency.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS: BANDPASS

LATEST REVISION: January 8, 1983 (Version 8.0)

3.6.3 Benioff

SUMMARY: Applies a Benioff filter to the data.

SYNTAX: BENIOFF

DESCRIPTION: This command is a digital approximation used to emulate the response of a short-period seismograph which was used by a VELA Program started by the U. S. Air Force about 1960. This Long Range Seismic Measurements (LRSM) program used truck vans and trailers to deploy moveable seismic systems, principally in North America, to record controlled source seismic experiments. Most of the seismic profiles were radial lines or circular arcs about the Nevada Test Site (NTS). Two semi-permanent sites or installations were Kanab, UT, and Mina, NV. LLNL continued operation of KN-UT and MI-NV after the LRSM program. These two stations used a variable-reluctance short-period seismometer (with a natural frequency of 1 Hz, critically damped) which was designed and named after Professor Hugo Benioff of Cal Tech. This short-period seismometer was coupled to a galvanometer (with a natural frequency of 5 Hz and damped to 0.9 critical). The coupling factor was nominally defined at 0.01 (or loosely coupled at low magnification settings which were used for recording the larger explosions) and the response was nearly flat-to-velocity between 1 and 5 Hz. When LLNL converted this system to a broadband, flat-to-velocity

telemetered system, an analog filter was designed to shape a passband into the LRSM short-period passband. This command executes a digital equivalent of that analog shaping filter which produces an output (measured in nanometers) analogous to the LRSM short-period system.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: May 15, 1987 (Version 10.2)

3.6.4 Convolve

SUMMARY: Compute the convolution of a master signal with itself and one or more other signals.

SYNTAX: CONVOLVE {MASTER name—n}, {NUMBER n},{LENGTH ON—OFF—v}, {TYPE RECTANGLE—HAMMING—HANNING—COSINE—TRIANGLE}

INPUT: MASTER name—n : Select master file in data file list by name or number. All files will be correlated against this one.

NUMBER n : Set number of correlation windows to be used.

LENGTH {ON} : Turn fixed window length option on.

LENGTH OFF : Turn fixed window length option off.

LENGTH v : Turn fixed window length option on and change window length in seconds to v.

TYPE RECTANGLE : Apply a rectangle function to each window. This is equivalent to applying no function to each window.

TYPE HAMMING : Apply a hamming function to each window.

TYPE HANNING : Apply a hanning function to each window.

TYPE COSINE : Apply a cosine function to each window.

TYPE TRIANGLE : Apply a triangle function to each window.

DEFAULT VALUES: CONVOLVE MASTER 1 NUMBER 1 LENGTH OFF TYPE RECTANGLE

DESCRIPTION: The convolve command allows the user to convolve a master signal with itself and one or many other signals. This command computes convolution, as approximated by $cv(\tau) = \int (f(t) g(\tau - t)) dt$, where int means summation over all time points with nonzero overlap. There is no $1/N$ normalization. This is very similar to cross correlation

which is defined by: $cc(\tau) = \int (f(t) g(t + \tau)) dt$. Syntax and output are similar to that of the CORRELATE command, except that the cross-correlation waveform is replaced by the convolution waveform.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ACKNOWLEDGEMENTS: This command is based on an algorithm developed by Dave Harris.

LATEST REVISION: Dec. 4, 1996 (Version 52a)

3.6.5 Correlate

SUMMARY: Computes the auto- and cross- correlation functions.

SYNTAX: CORRELATE {MASTER name—n}, {NUMBER n},{LENGTH ON—OFF—v}, {TYPE RECTANGLE—HAMMING—HANNING—COSINE—TRIANGLE}

INPUT: MASTER name—n : Select master file in data file list by name or number. All files will be correlated against this one.

NUMBER n : Set number of correlation windows to be used.

LENGTH {ON} : Turn fixed window length option on.

LENGTH OFF : Turn fixed window length option off.

LENGTH v : Turn fixed window length option on and change window length in seconds to v.

TYPE RECTANGLE : Apply a rectangle function to each window. This is equivalent to applying no function to each window.

TYPE HAMMING : Apply a hamming function to each window.

TYPE HANNING : Apply a hanning function to each window.

TYPE COSINE : Apply a cosine function to each window.

TYPE TRIANGLE : Apply a triangle function to each window.

DEFAULT VALUES: CORRELATE MASTER 1 NUMBER 1 LENGTH OFF TYPE RECTANGLE

DESCRIPTION: An auto-correlation function is computed on the signal which you declare to be the master one, and a cross-correlation function is calculated between it and each of the other signals in memory. The windowing features of this command allow you to compute an average correlation function over a set of data windows. The number of windows is

selectable and there are five standard windowing functions to choose from. When this windowing feature is on, a cross-correlation function is computed for each window. This collection of cross-correlation functions is then averaged, cut to the same length as the original data file, and replaces the data file in memory. You may also select the length of each window. Window overlap is automatically calculated and used whenever the product of the requested window length (LENGTH option) and the number of windows (NUMBER option) exceeds the number of points in the data file (NPTS). By default, this windowing feature is off.

EXAMPLES: To calculate the correlation functions using the third file in memory **as the master file:**

u: CORRELATE MASTER 3

You could also specify the master file by name if this is easier. Assume you have two data files that each contain 1000 points of noise. To compute the average correlation functions using 10 windows of 100 points **each (i.e. no overlap) with a hanning function applied to each window:**

u: CORRELATE TYPE HANNING NUMBER 10

To achieve a twenty percent overlap of each window, set the window length to the equivalent of 120 data points. Assuming a sampling interval of 0.025 (40 **samples per second**) **this would be three seconds as shown below:**

u: CORRELATE TYPE HANNING NUMBER 10 LENGTH 3.0

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ACKNOWLEDGEMENTS: This command is based on an algorithm developed by Dave Harris.

LATEST REVISION: Dec. 4, 1996 (Version 52a)

3.6.6 Dft

3.6.7 Divomega

SUMMARY: Performs integration in the frequency domain.

DESCRIPTION: This command divides each point of a spectral file by its frequency **given by:**

$$\text{OMEGA} = 2.0 * \text{PI} * \text{FREQ}$$

This is analogous to integrating the equivalent time series file. The spectral file can in either amplitude-phase or real-imaginary format. This is often convenient with normal data but is critical when obtaining the FFT of data whose spectra ranges over many orders of magnitude. For example, suppose you have prewhitened a data file by using the DIF command, and then taken its transform using the FFT command. The effect of the differentiation in time domain can be removed by an integration in the frequency domain using this command.

EXAMPLES: The steps discussed above are shown in this example:

u: READ FILE1

u: DIF

u: FFT AMPH

u: DIVOMEGA

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS: DIF, FFT, MULOMEGA, WRITESP, READSP

LATEST REVISION: May 15, 1987 (Version 10.2)

3.6.8 Envelope

SUMMARY: Computes the envelope function using a Hilbert transform.

SYNTAX: ENVELOPE

DESCRIPTION: This command computes the envelope function of the data in memory. The **envelope is defined by:** where $x(n)$ is the original signal and $y(n)$ its Hilbert transform (see HILBERT.) As with HILBERT, very long period data should be decimated (see DECIMATE) prior to processing.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS: HILBERT, DECIMATE

ACKNOWLEDGEMENT: The subroutines used to perform the Hilbert transform were designed and developed by Dave Harris.

LATEST REVISION: April 21, 1989 (Version 10.4c)

3.6.9 Filterdesign

SUMMARY: Produces a graphic display of a filter's digital vs. analog characteristics **for:** amplitude, phase, and impulse response curves, and the group delay.

SYNTAX: FILTERDESIGN [PRINT [pname]] [FILE [prefix]][filteroptions] [delta]

where filteroptions are the same as those used in the various filter commands in SAC, including the filter type. delta is the sampling interval of the data.

*** Note that the order of options is important. If the PRINT option is used, it must be the first option. If the FILE option is used, it must precede the filter options.

PRINT {pname} : Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. ****Note:** this must be the first option given on the command line. (This makes use of the SGF capability.)

FILE {prefix} : Writes three SAC files to disk. These files contain the digital responses determined in **the FILTERDESIGN:** [prefix].spec is of type IAMPH, and contains both the amplitude and phase information from the FILTERDESIGN. [prefix].gd is of type ITIME, and contains the group delay information from the FILTERDESIGN. *** Note that in spite of the fact that the file is of type ITIME, group delay is a function of frequency. It is incumbent upon the user to remember that even though the plots will have seconds for units, the actual units are hertz. [prefix].imp is of type ITIME, and contains the impulse response.

In each of these SAC files, the user header fields are set **as follows:**

user0 pass code 1) low pass 2) high pass 3) band pass 4) band reject
user1 type code 1) Butterworth 2) Bessel 3) C1 4) C2 user2 number of poles

user3 number of passes user4 tranbw user5 attenuation user6 delta user7
first corner user8 second corner if present, or -12345 if not

kuser0 pass (lowpass, highpass, bandpass, or bandrej) kuser1 type (But-
ter, Bessel, C1, or C2)

DEFAULT VALUES: Only the delta parameter has a default (0.025
seconds). Options for filter type and related parameters must be supplied.

DESCRIPTION: The FILTERDESIGN command is implemented through
XAPiir, a basic recursive digital filtering package (see REFERENCES).
XAPiir implements the standard recursive digital filter design through bilin-
ear transformation of prototype analog filters. These prototype filters, spec-
ified in terms of poles and zeros, are then transformed to highpass, bandpass
and band reject filters using analog spectral transformations. FILTERDE-
SIGN displays digital filter responses as solid lines and analog responses as
dashed lines. On color monitors, digital curves are blue while analog curves
are amber.

EXAMPLES: The following example shows how the FILTERDESIGN
command is used to produce the digital and analog response curves for a
highpass, 2 Hz., six pole, two pass filter on data with a sampling rate of .025
seconds.

```
fd hp c 2 n 6 p 2 delta .025
```

**SEE COMMANDS: HIGHPASS, LOWPASS, BANDPASS, BAN-
DREJECT UCRL-ID-106005. XAPiir:** A Recursive Digital Filtering
Package. David Harris. September 21, 1990 In Xwindows, a linestyle prob-
lem may cause both analog and digital traces to plot as solid lines.

LATEST REVISION: July 22, 1991 (Version 0.58)

3.6.10 Fir

SUMMARY: Applies a finite-impulse-response filter.

SYNTAX: FIR {REC—FFT},file

INPUT: FFT : Apply the FIR filter using the transform method.

REC : Apply the FIR filter recursively.

file : The name of the file containing the FIR filter.

ALTERNATE FORMS: DFT may be used in place of FFT.

DEFAULT VALUES: FIR FFT FIR

DESCRIPTION: The filter applied by this command must have been designed by using the DFIR interactive filter design program (see BUGS below). The filter is applied using the transform method unless you request the recursive method or the number of data points is too large for the transform method. These filters all have zero phase distortion but can produce precursors with impulsive signals.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

1601: File and filter sampling intervals not equal for

- The filter must be designed using the same sampling rate as the data to be filtered. **1603:** Inadequate memory to perform FIR filter.

WARNING MESSAGES: 1602: Inadequate memory to perform FIR filter using DFT.

- the recursive method will be used automatically.

LIMITATIONS: Maximum number of data points for transform method is 4096. The DFIR routine has since vanished, as this has not been used by the Seismologists at LLNL for some years. 1. See the author for information on the use of DFIR. 2. See Chapter 3 of Rabiner and Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, 1975 for a discussion of FIR filters.

LATEST REVISION: July 22, 1991 (Version 8.0)

3.6.11 Hanning

SUMMARY: Applies a “hanning” window to each data file.

SYNTAX: HANNING

DESCRIPTION: The “hanning” window is a recursive smoothing algorithm defined at each **interior data point, j, as:**

$$Y(j) = 0.25*Y(j-1) + 0.50*Y(j) + 0.25*Y(j+1)$$

Each end point is set equal to its closest interior point.

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN Blackman and Tukey, “The Measurement of Power Spectra”, Dover Publications, New York, 1958.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.6.12 Highpass

SUMMARY: Applies an IIR highpass filter.

SYNTAX: HIGHPASS {BUTTER—BESSEL—C1—C2},{CORNERS v1 v2}, {NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}

INPUT: BUTTER : Apply a Butterworth filter.

BESSEL : Apply a Bessel filter.

C1 : Apply a Chebyshev Type I filter.

C2 : Apply a Chebyshev Type II filter.

CORNER v : Set corner frequency to v.

NPOLES n : Set number of poles to n {range: 1-10}.

PASSES n : Set number of passes to n {range: 1-2}.

TRANBW v : Set the Chebyshev attenuation factor to v.

ATTEN v : Set the Chebyshev attenuation factor to v.

DEFAULT VALUES: HIGHPASS BUTTER CORNER 0.2 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.

DESCRIPTION: See the BANDPASS command for definitions of the filter parameters and descriptions on how to use them.

EXAMPLES: To apply a four-pole Butterworth with a corner at 2 Hz.

u: HIGHPASS NPOLES 4 CORNER 2

To apply a two-pole two-pass Bessel with the same corner.

u: HP N 2 BE P 2

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

1002: Bad value for

- corner frequency larger than Nyquist frequency.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS: BANDPASS

LATEST REVISION: January 8, 1983 (Version 8.0)

3.6.13 Hilbert

SUMMARY: Applies a Hilbert transform.

SYNTAX: HILBERT

DESCRIPTION: Each data file, $y(n)$, in the data file list is replaced by its Hilbert transform, $x(n)$. The transform is found by convolving $y(n)$ (in the time **domain**) **with a 201 point FIR filter:** The filter impulse response is obtained by windowing an ideal Hilbert transformer impulse response with a Hamming **window:** In the frequency domain, this filter approximates the transfer **function:** The phase criterion is met exactly (90 degree phase shift at each frequency), and the magnitude response (ideally unity) is shown on the next page. Note that the operation is inexact in small regions about DC and the folding frequency. If transforms are to be taken of very low frequency data, such as long period surface waves, the signals should first be decimated. Since the transformation is performed in the time domain, computations are done in-place using the overlap-save algorithm. There are no restrictions on the length of data file.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ACKNOWLEDGEMENT: The subroutines used to perform the Hilbert transform were designed and developed by Dave Harris.

LATEST REVISION: April 21, 1989 (Version 10.4c) Amplitude Response of Hilbert Transform.

3.6.14 Idft

3.6.15 Keepam

SUMMARY: Keep amplitude component of spectral files (of either the

AMPH or RLIM format) in SAC memory.

SYNTAX: KEEPAM

DESCRIPTION: This command is an easy way for users to drop the phase component, so that the amplitude data may be subjected to algebraic operations which require single-dimensional data. If the files exist in the RLIM format, the data is first converted to the AMPH format, before phase is dropped. The resulting data files containing the amplitude component will exist as generic xy files, so that they can be distinguished for time-domain files. May 28, 1991 (Version 10.5c)

3.6.16 Khronhite

SUMMARY: Applies a Khronhite filter to the data.

SYNTAX: KHRONHITE {v}

INPUT: v : Cutoff frequency in hertz.

DEFAULT VALUES: KHRONHITE 2.0

DESCRIPTION: This lowpass filter is a digital approximation of an analog filter which was a cascade of two fourth-order Butterworth lowpass filters. This lowpass filter has been used with a corner frequency of 0.1 Hz to enhance measurements of the amplitudes of the fundamental mode Rayleigh wave (Rg) at regional distances.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: February 15, 1987

3.6.17 Lowpass

SUMMARY: Applies an IIR lowpass filter.

SYNTAX: LOWPASS {BUTTER—BESSEL—C1—C2},{CORNER v},
{NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}

INPUT: BUTTER : Apply a Butterworth filter.

BESSEL : Apply a Bessel filter.

C1 : Apply a Chebyshev Type I filter.

C2 : Apply a Chebyshev Type II filter.

CORNER v : Set corner frequency to v.

NPOLES n : Set number of poles to n {range: 1-10}.

PASSES n : Set number of passes to n {range: 1-2}.

TRANBW v : Set the Chebyshev transition band width to v.

ATTEN v : Set the Chebyshev attenuation factor to v.

DEFAULT VALUES: LOWPASS BUTTER CORNER 0.4 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.

DESCRIPTION: See the BANDPASS command for definitions of the filter parameters and descriptions on how to use them.

EXAMPLES: To apply a four-pole Butterworth with a corner at 2 Hz.

u: LOWPASS NPOLES 4 CORNER 2

To apply a two-pole two-pass Bessel with the same corner.

u: LP N 2 BE P 2

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

1002: Bad value for

- corner frequency larger than Nyquist frequency. See Chapter 4 of Rabiner and Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, 1975 for a discussion of IIR filters.

SEE COMMANDS: BANDPASS

LATEST REVISION: January 8, 1983 (Version 8.0)

3.6.18 Mulomega

SUMMARY: Performs differentiation in the frequency domain.

DESCRIPTION: This command multiplies each point of a spectral file by its frequency given **by:**

$\text{OMEGA} = 2.0 * \text{PI} * \text{FREQ}$

This is analogous to differentiating the equivalent time series file. The spectral file can in either amplitude-phase or real-imaginary format.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: May 15, 1987 (Version 10.2)

3.6.19 Plotsp

SUMMARY: Plots spectral data in several different formats.

SYNTAX: PLOTSP {type},{mode} **where type is one of the following:**

ASIS—RLIM—AMPH—RL—IM—AM—PH

where mode is one of the following:

LINLIN—LINLOG—LOGLIN—LOGLOG

INPUT: ASIS : Plot components in their present format.

RLIM : Plot real and imaginary components.

AMPH : Plot amplitude and phase components.

RL : Plot real component only.

IM : Plot imaginary component only.

AM : Plot amplitude component only.

PH : Plot phase component only.

LINLIN : Set x-y scaling mode to linear-linear.

LINLOG : Set x-y scaling mode to linear-logarithmic.

LOGLIN : Set x-y scaling mode to logarithmic-linear.

LOGLOG : Set x-y scaling mode to logarithmic-logarithmic.

DEFAULT VALUES: PLOTSP ASIS LOGLOG

DESCRIPTION: SAC data files may contain either time-series data or spectral data. Certain fields in the header distinguish between the two formats. Most plot commands (PLOT, PLOT1, etc.) only plot time-series data. This command lets you plot spectral data. You may plot one or both spectral components using this command. One frame is generated for each spectral component plotted. Other plot formats will be added as needed. You can also select the scaling mode to be used. This scaling mode applies only to PLOTSP.

EXAMPLES: To get a logarithmic-linear plot of the spectral amplitude of a data file:

u: READ FILE1

u: FFT

u: PLOTSP AM LOGLIN

ERROR MESSAGES: 1301: No data files read in.

1305: Illegal operation on time series file

LATEST REVISION: May 15, 1987 (Version 10.2)

3.6.20 Readsp

SUMMARY: Reads spectral files written by WRITESP and WRITESPE.

SYNTAX: READSP {AMPH—RLIM—SPE} {filelist}

INPUT: RLIM : Read real and imaginary components.

AMPH : Read amplitude and phase components.

SPE : Read spectral estimation subprocess files. The data is converted from power to amplitude. The phase component is set to zeros.

filelist : A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DEFAULT VALUES: READSP AMPH

DESCRIPTION: The WRITESP command writes each spectral data component to disk as a separate file. You may then process each component separately. This command lets you reconstruct the spectral data from the two components. See the WRITESP documentation for more details. The SPE option allows you to read in and convert to spectral format, files that were written using the WRITESPE command in the Spectral Estimation Subprocess. This allows you to use commands such as MULOMEGA and DIVOMEGA on these spectral estimates.

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred from the SAC data buffer to the CSS data buffer. When READSP is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

EXAMPLES: See the example in the WRITESP documentation.

SEE COMMANDS: WRITESP

REFERENCES: Spectral Estimation Subprocess Manual

LATEST REVISION: April 21, 1989 (Version 10.4c)

3.6.21 Unwrap

SUMMARY: Computes amplitude and unwrapped phase.

SYNTAX: UNWRAP {FILL {ON—OFF—n}}, {INTTHR v}, {PVTHR v}

INPUT: FILL {ON} : Turn zero fill option on.

FILL OFF : Turn zero fill option off.

FILL n : Turn zero fill option on and change fill value to n.

INTTHR v : Change the integration threshold constant to v.

PVTHR v : Change the principal value threshold constant to v.

DEFAULT VALUES: UNWRAP FILL OFF INTTHR 1.5 PVTHR 0.5

DESCRIPTION: This command transforms time-series data in memory to spectral data containing amplitude and “unwrapped” phase components. This procedure works for data with a “smoothly varying phase.” The data is filled with zeros to the next power of two before being transformed. You may specify a larger number of zeros by using the FILL option. This is an implementation of the algorithm due to Tribolet. Two methods are used to estimate the unwrapped phase at each frequency. One is to numerically integrate the phase derivative through the use of the fast Fourier transform. The step size used in this trapezoidal integration is halved at each frequency if necessary to obtain a consistent estimate. You can control the threshold value on this check using the INTTHR option. This value is in radians. Decreasing INTTHR will improve the phase estimate. Too small a value, however, will not allow the solution to converge. The second method used in this algorithm is to first compute the principle value of the phase using the inverse tangent function. The unwrapped phase is estimated by adding multiples of 2π to the principal value until the discontinuities are reduced to values less than a threshold value. You control the threshold value on this check using the PVTHR option. Again, decreasing this threshold value

will improved the phase estimate, but will also increase the chance that no solution may be found. **Initial trial values for these two thresholds are usually such that:**

$PI/4$; PVTHR ; INTTHR ; $2*PI$

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1606: Maximum allowable DFT is

- Too many data points in file.

WARNING MESSAGES: 1610: Unwrap failed at data point for file

- Adjust threshold constants and retry.

HEADER CHANGES: B, E, and DELTA are changed to the beginning, ending and sampling frequencies of the transform respectively. The original values of B, E, and DELTA are saved as SB, SE, and SDELTA and are restored when an inverse transform is performed.

LIMITATIONS: The maximum length of data that can currently be transformed is 4096. Tribolet, Jose M.; “A New Phase Unwrapping Algorithm”; IEEE Transactions on Acoustics, Speech, and Signal Processing; Vol. ASSP-25, No 2, April 1977; page 170.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.6.22 Wiener

SUMMARY: Designs and applies an adaptive Wiener filter.

SYNTAX: WIENER {[W]INDOW pdw} {[N]COEFF n} {MU OFF — ON — v} {[EPS]ILON OFF — ON — e}

INPUT: WINDOW pdw : Set filter design window to pdw. A partial data window which consists of a start time and stop time. These times can be absolute ones or ones relative to certain header fields. See the CUT command for details on pdw.

NCOEFF n : Set the number of filter coefficients to n.

MU off — on — v : Set the adaptation step size parameter. Off sets mu to zero. On sets $\mu = 1.95 / \text{Rho}(0)$. Where $\text{Rho}(0)$ is the autocorrelation in pdw at zero lag. v sets $\mu = v$.

EPSILON e : Set ridge regression parameter to epsilon. Can help stabilize the wiener filter by increasing the diagonal elements of the autocorrelation matrix by epsilon. When epsilon is ON, SAC will use the value entered by the user (or zero if no value was entered). When epsilon is OFF, SAC will loop through the following increasing values of epsilon (0.0, 1e-5, 1e-4, 1e-3, 1e-2), until the wiener filter is stable, or until the list has been exhausted. If epsilon == 0 does not work, SAC will produce one or more warning messages. If none of the values work, SAC will produce an error message.

DEFAULT VALUES: WIENER WINDOW B 0 10 NCOEFF 30 MU OFF EPSILON OFF

DESCRIPTION: A prediction error filter is designed using the Yule-Walker Method from an autocorrelation function estimated from the designated partial data window. This window can be any portion of the file. The filter is then applied to the entire signal, i.e. the signal is replaced by the residual error sequence. This filter may be used as a prewhitener or as a detection preprocessor for transient signals. The filter can be made adaptive in time by specifying a non-zero value for MU. Large values of MU may cause instability.

EXAMPLES: The following command would apply a non-adaptive filter, with the first ten **seconds being the design window**:

u: WIENER WINDOW B 0 10 MU 0.

The following command would apply a filter with 40 coefficients, with a design window from the beginning of the file to 1 second before the first **arrival**:

u: WIENER NCOEFF 40 WINDOW B A -1

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

1608: Bad Wiener filter noise window for file

- Filter design window does not lie within file window. - Make sure header fields used in window are defined..

WARNING MESSAGES: 1609: Numerical instability in Wiener filter for file

1614: Numerical instability in Wiener; will retry with $\epsilon = e$ where e denotes the next value of epsilon to be tried.

- The filtered data may or may not be incorrect.

SEE COMMANDS: CUT

LATEST REVISION: March 12, 1997 (Version 00.53)

3.6.23 Writesp

SUMMARY: Writes spectral files to disk as “normal” data files.

SYNTAX: WRITESP {type} {COMMIT—ROLLBACK—RECALLTRACE}
{OVER—filelist} **where type is one of the following:** ASIS—RLIM—AMPH—RL—IM—AM—PH

INPUT: ASIS : Write components in their present format.

RLIM : Write real and imaginary components.

AMPH : Write amplitude and phase components.

RL : Write real component only.

IM : Write imaginary component only.

AM : Write amplitude component only.

PH : Write phase component only.

COMMIT : Commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to writing files. COMMIT is the default.

ROLLBACK : reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE : - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

filelist : A list of SAC binary data files. This list may contain simple filenames and full or relative pathnames.

DEFAULT VALUES: WRITESP ASIS COMMIT

DESCRIPTION: SAC data files may contain either time-series data or spectral data. Certain fields in the header distinguish between the two formats. When you read (see READ) a time-series file into memory, take the fast fourier transform (see FFT), and write the data to disk (see WRITE), then the data on disk will be in the spectral format. Certain operations can only be performed on time-series data and certain operations only on spectral data. For example, you can't apply a taper to spectral data files or multiply two spectral files together. This is a protection mechanism built into SAC. Sometimes, however, you may need to perform some of these operations on spectral data. To override SAC's protection mechanism, you can use this command to write spectral data to disk as time-series data. Each component is written as a separate data file. You may then read these files back into SAC and perform any operation that you wish, since SAC thinks they are time series data files. Once these calculations are completed, you may write the modified data back to disk using the WRITE command. If you wish to reconstruct the spectral data file, use the READSP command. To help you keep track of the data on disk, SAC appends a suffix to the filename you request that identifies the spectral component stored in that file. The suffixes are ".RL", ".IM", ".AM", and ".PH" for the real component, imaginary component, amplitude, and phase respectively.

EXAMPLES: Assume that you want to perform some operations on the spectral amplitude of **FILE1**:

```
u: READ FILE1
u: FFT AMPH
u: WRITESP OVER
```

SAC will then write out two files, FILE1.AM and FILE1.PH. Now you perform the **operations on the amplitude file**:

```
u: READ FILE1.AM
u: ...perform operations.
u: WRITE OVER
```

Now the files on disk represent the modified spectral data. If you wanted to **reconstruct the SAC spectral data file and take the inverse transform**:

u: READSP FILE1

u: IFFT

u: WRITE FILE2

Note: for examples of the behavior of COMMIT, ROLLBACK, and RECALLTRACE, see the commands of the same name.

ERROR MESSAGES: 1301: No data files read in.

1305: Illegal operation on time series file

HEADER CHANGES: B, E, and DELTA for the files on disk will contain the beginning, ending, and incremental frequency in Hz.

SEE COMMANDS: READSP, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.7 UOM: Unary Operations Module

3.7.1 Abs

3.7.2 Add

SUMMARY: Adds a constant to each data point.

SYNTAX: ADD {v1 {v2 ... vn} }

INPUT: **v1** : Constant to add to first file.

v2 : Constant to add to second file.

vn : Constant to add to nth file.

DEFAULT VALUES: ADD 0.0

DESCRIPTION: This command will add a constant to each element of each data file in memory. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files.

EXAMPLES: To add 5.1 to each element of F1 and 6.2 to each element of F2 and F3:

u: READ F1 F2 F3

u: ADD 5.1 6.2

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 8, 1983 (Version 8.0)

3.7.3 Dif

SUMMARY: Differentiates data in memory.

SYNTAX: DIF {TWO—THREE—FIVE}

INPUT: TWO : Apply a two point difference operator.

THREE : Apply a three point difference operator.

FIVE : Apply a five point difference operator.

DEFAULT VALUES: DIF TWO

DESCRIPTION: The two-point algorithm is: $OUT(J) = \frac{(DATA(J+1) - DATA(J))}{\{DELTA\}}$ The last output point is not defined by this

algorithm. It is also not a centered algorithm. SAC takes care of these problems by decreasing the number of points in the file (NPTS) by one and by increasing the begin time (B) by half the sampling interval (DELTA).

The three-point (centered two-point) algorithm is: $OUT(J) = \frac{(DATA(J+1) - DATA(J-1))}{\{2\} \{DELTA\}}$

The first and last output point is not defined by this algorithm. SAC decreases NPTS by 2 and increases B by DELTA.

The five-point (centered four-point) algorithm is: $OUT(J) = \frac{(DATA(J+2) - DATA(J-2))}{\{4\} \{DELTA\}}$

The first two and last two output points are not defined by this algorithm. SAC applies the three-point operator to the second points from each end, decreases NPTS by 2, and increases B by DELTA.

The five-point (centered four-point) algorithm is: $OUT(J) = \frac{(DATA(J+2) - DATA(J-2))}{\{4\} \{DELTA\}}$

The first two and last two output points are not defined by this algorithm. SAC applies the three-point operator to the second points from each end, decreases NPTS by 2, and increases B by DELTA.

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

HEADER CHANGES: NPTS, B, E, DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 15, 1985 (Version 9.10)

3.7.4 Div

SUMMARY: Divides each data point by a constant.

SYNTAX: DIV {v1 {v2 ... vn} }

INPUT: **v1** : Constant to divide first file by.

v2 : Constant to divide second file by.

vn : Constant to divide nth file by.

DEFAULT VALUES: DIV 1.

DESCRIPTION: This command will divide each element of each data file in memory by a constant. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files in memory.

EXAMPLES: To divide each element of F1 by 5.1 and each element of F2 and F3 by 6.2:

u: READ F1 F2 F3

u: DIV 5.1 6.2

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

1701: Can't divide by zero.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.7.5 Exp

SUMMARY: Computes the exponential of each data point.

SYNTAX: EXP

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 15, 1985 (Version 9.10)

3.7.6 Exp10

SUMMARY: Computes the base 10 exponential ($10.^{*}y$) of each data point.

SYNTAX: EXP10

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 15, 1985 (Version 9.10)

3.7.7 Int

SUMMARY: Performs integration using the trapezoidal or rectangular rule.

SYNTAX: INT TRAPEZOIDAL — RECTANGULAR

DEFAULT VALUES: INT TRAPEZOIDAL

DESCRIPTION: This command uses the trapezoidal or rectangular integration method. The first output data point is set to zero. If the trapezoidal option is used, the number of points is reduced by one. The data does not have to be evenly spaced.

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMIN

LATEST REVISION: March 20, 1992 (Version 10.6e)

3.7.8 Log

SUMMARY: Takes the natural logarithm of each data point.

SYNTAX: LOG

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

1340: data points outside allowed range contained in file

- All data points must be positive.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 15, 1985 (Version 9.10)

3.7.9 Log10

SUMMARY: Takes the base 10 logarithm of each data point.

SYNTAX: LOG10

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

1340: data points outside allowed range contained in file

- All data points must be positive.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 15, 1985 (Version 9.10)

3.7.10 Mul

SUMMARY: Multiplies each data point by a constant.

SYNTAX: MUL {v1 {v2 ... vn} }

INPUT: v1 : Constant to multiply first file by.

v2 : Constant to multiply second file by.

vn : Constant to multiply nth file by.

DEFAULT VALUES: MUL 1.

DESCRIPTION: This command will multiply each element of each data file in memory by a constant. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files in memory.

EXAMPLES: To multiply each element of F1 by 5.1 and each element of F2 and F3 by 6.2:

u: READ F1 F2 F3

u: MUL 5.1 6.2

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

LATEST REVISION: January 8, 1983 (Version 8.0)

3.7.11 Sqr

SUMMARY: Squares each data point.

SYNTAX: SQR

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 8, 1983 (Version 8.0)

3.7.12 Sqrt

SUMMARY: Takes the square root of each data point.

SYNTAX: Sqrt

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

1702: Non-positive values found in file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 8, 1983 (Version 8.0)

3.7.13 Sub

SUMMARY: Subtracts a constant from each data point.

SYNTAX: SUB {v1 {v2 ... vn} }

INPUT: v1 : Constant to subtract from first file.

v2 : Constant to subtract from second file.

nv : Constant to subtract from nth file.

DEFAULT VALUES: SUB 0

DESCRIPTION: This command will subtract a constant from each element of each data file currently in memory. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files in memory.

EXAMPLES: To subtract 5.1 from each element of F1 and 6.2 from each element of F2 and **F3:**

u: READ F1 F2 F3

u: SUB 5.1 6.2

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

LATEST REVISION: January 8, 1983 (Version 8.0)

3.8 BOM: Binary Operations Module

3.8.1 Addf

SUMMARY: Adds a set of data files to data in memory.

SYNTAX: ADDF {NEWHDR ON—OFF} filelist

INPUT: NEWHDR ON—OFF : By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist : A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION: This command can be used to add a single file to a set of files or to add one set of files to another set. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of data points. These last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES: To add one file to three other files:

u: READ FILE1 FILE2 FILE3

u: ADDF FILE4

To add two files to two other files:

u: READ FILE1 FILE2

u: ADDF FILE3 FILE4

HEADER CHANGES: If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1803: No binary data files read in.

1307: Illegal operation on spectral file

1306: Illegal operation on unevenly spaced file

1801: Header field mismatch:

- sampling interval or number of points are not equal. - can be controlled using the BINOPERR command.

WARNING MESSAGES: 1802: Time overlap:

- the file addition is still performed.

SEE COMMANDS: READ, BINOPERR

LATEST REVISION: May 26, 1999 (Version 0.58)

3.8.2 Binoperr

SUMMARY: Controls errors that can occur during binary file operations.

SYNTAX: BINOPERR {NPTS FATAL—WARNING—IGNORE}, {DELTA FATAL—WARNING—IGNORE}

INPUT: NPTS : Change error condition for unequal number of data points.

DELTA : Change error condition for unequal sampling intervals.

FATAL : Make error condition fatal. Control is immediately returned to the user's terminal. Additional commands typed on the same line or in the same command file are ignored.

WARNING : Send a warning message to the user. Correct the error condition and continue.

IGNORE : Correct the error condition and continue.

DEFAULT VALUES: BINOPERR NPTS FATAL DELTA FATAL

DESCRIPTION: SAC checks for certain common errors whenever you execute a binary operations module command (ADDF, DIVF, etc.) Using this command, you can control what SAC does when it finds one of these errors. If you make an error condition fatal, then SAC will stop executing

the current command, will ignore all commands in its queue, will print an error message to the terminal, and will return control to you. If you make an error condition a warning, then SAC will send you a warning message, correct the condition as best it can, and continue. If you tell SAC to ignore a condition, then SAC will correct the condition and continue without telling you the condition even occurred. One of these error conditions occurs when the number of data points in the two files to be operated on are not equal. Corrective action in this case is to perform the operation using the number of data points in the smaller file. Another error condition occurs when the sampling intervals of the two files are not the same. The corrective action in this case is to use the sampling interval of the first data file.

EXAMPLES: Assume that FILE1 has 1000 data points and FILE2 has 950 data points.

u: BINOPERR NPTS FATAL

u: READ FILE1

u: ADDF FILE2

u: ERROR: Header field mismatch: NPTS FILE1 FILE2

The file addition was not performed. Assume you now type:

u: BINOPERR NPTS WARNING

u: ADDF FILE2

u: WARNING: Header field mismatch: NPTS FILE1 FILE2

The file addition was performed on the first 950 data points of each file.

SEE COMMANDS: ADDF, SUBF, MULF, DIVF

LATEST REVISION: January 8, 1983 (Version 8.0)

3.8.3 Divf

SUMMARY: Divides data in memory by a set of data files.

SYNTAX: DIVF {NEWHDR ON—OFF} filelist

INPUT: NEWHDR ON—OFF : By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist : A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION: This command can be used to divide a set of files by a single file or by another set of files. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of points. This last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES: To divide three files by a single file:

u: READ FILE1 FILE2 FILE3

u: DIVF FILE4

To divide two files by two other files:

u: READ FILE1 FILE2

u: DIVF FILE3 FILE4

HEADER CHANGES: If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1803: No binary data files read in.

1307: Illegal operation on spectral file

1306: Illegal operation on unevenly spaced file

1801: Header field mismatch:

- either the sampling interval or the number of points are not equal. -
can be controlled using the BINOPERR command.

WARNING MESSAGES: 1802: Time overlap:

- the file division is still performed.

SEE COMMANDS: READ, BINOPERR

LATEST REVISION: May 26, 1999 (Version 0.58)

3.8.4 Merge

SUMMARY: Merges (concatenates) a set of files to data in memory.

SYNTAX: MERGE {COMMIT—ROLLBACK—RECALLTRACE} {filelist}

INPUT: COMMIT : Commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to merging files. COMMIT is the default.

ROLLBACK : reverts to the last committed version of the header and waveform before merging files.

RECALLTRACE : - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

filelist : A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION: The data in the files in this merge list is appended or concatenated to the data in memory. Each pair of files to be merged is checked to make sure they have the same sampling interval and station name. The end time of the first file is also compared with the begin time of the second file. If there is a gap in these times, a warning message is generated and the appropriate number of zeros is placed in the merged file. If there is an overlap in these times, an error message is generated and no merge is performed.

EXAMPLES: To merge FILE3 and FILE4 to FILE1 and FILE2 respectively:

u: READ FILE1 FILE2

u: MERGE FILE3 FILE4

To merge files for the same station, say ELKO.Z, from four different events **each stored in a separate UNIX directory:**

u: READ /data/event1/elko.z

u: MERGE /data/event2/elko.z

u: MERGE /data/event3/elko.z

u: MERGE /data/event4/elko.z

If you were on different kind of computer (i.e. not UNIX) then only the way you specify full pathnames would be different.

Note: for examples of the behavior of COMMIT, ROLLBACK, and RECALLTRACE, see the commands of the same name.

HEADER CHANGES: NPTS, DEPMIN, DEPMAX, DEPMEN, E

ERROR MESSAGES: 1301: No data files read in.

1803: No binary data files read in.

1307: Illegal operation on spectral file

1306: Illegal operation on unevenly spaced file

1801: Header field mismatch:

- either the sampling interval or station name are not equal. **1802:**

Time overlap:

WARNING MESSAGES: 1805: Time gap (zeros added):

SEE COMMANDS: READ, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: Oct. 27, 1998 (Version 0.58)

3.8.5 Mulf

SUMMARY: Multiplies a set of files by the data in memory.

SYNTAX: MULF {NEWHDR ON—OFF} filelist

INPUT: NEWHDR ON—OFF : By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist : A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION: This command can be used to multiply a single file by a set of files or to multiply one set of files by another set. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of points. This last two restrictions can be eliminated using the BINOPERR command. If there are

more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES: To multiply three files by a single file:

u: READ FILE1 FILE2 FILE3

u: MULF FILE4

To multiply two files by two other files:

u: READ FILE1 FILE2

u: MULF FILE3 FILE4

HEADER CHANGES: If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN **1301:** No data files read in.

1803: No binary data files read in.

1307: Illegal operation on spectral file

1306: Illegal operation on unevenly spaced file

1801: Header field mismatch:

- either the sampling interval or the number of points are not equal. -
can be controlled using the BINOPERR command.

WARNING MESSAGES: 1802: Time overlap:

- the file multiplication is still performed.

SEE COMMANDS: READ, BINOPERR

LATEST REVISION: May 26, 1999 (Version 0.58)

3.8.6 Subf

SUMMARY: Subtracts a set of data files from data in memory.

SYNTAX: SUBF {NEWHDR ON—OFF} filelist

INPUT: NEWHDR ON—OFF : By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist : A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION: This command can be used to subtract a single file from a set of files or to subtract one set of files from another set. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of points. This last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES: To subtract one file from three other files:

u: READ FILE1 FILE2 FILE3

u: SUBF FILE4

To subtract two files from two other files:

u: READ FILE1 FILE2

u: SUBF FILE3 FILE4

HEADER CHANGES: If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1803: No binary data files read in.

1307: Illegal operation on spectral file

1306: Illegal operation on unevenly spaced file

1801: Header field mismatch:

- either the sampling interval or the number of points are not equal. -
can be controlled using the BINOPERR command.

WARNING MESSAGES: 1802: Time overlap:

- the file subtraction is still performed.

SEE COMMANDS: READ, BINOPERR

LATEST REVISION: May 26, 1999 (Version 0.58)

3.9 EAM: Event Analysis Module

3.9.1 Apk

SUMMARY: Applies an automatic event picking algorithm.

SYNTAX: APK {param v {param v} ... },{VALIDATION ON—OFF}

INPUT: **param v** : Define a new value for one of the pick parameters.

param : C1—C2—C3—C4—C5—C6—C7—C8—D5—D8—D9—I3—I4—I6.

,BREAK These parameters are defined below.

VALIDATION ON : Turn validation phase on.

VALIDATION OFF : Turn validation phase off.

DEFAULT VALUES: APK C1 0.985 C2 3.0 C3 0.6 C4 0.03 C5 5.0
C6 0.0039 C7 100. C8 -0.1 D5 2. D8 3. D9 1. I3 3 I4 40 I6 3 VALIDATION
ON

DESCRIPTION: The algorithm used in this automatic picker was originally obtained from the USGS in Menlo Park and is based upon work by Rex Allen (see reference below.) The detection of a pick is based upon abrupt changes in the ratio of a short term and long term running average of the signal. Once detected, the pick is subjected to an optional validation phase which attempts to distinguish a true event from cultural noise. Once validated, the pick is further evaluated to determine other characteristics of the event. Currently this is limited to its duration. Other features such as maximum amplitude, period, and decay rate may be added as required. Most of the parameters in this command need never be changed. They are available if the user wishes to fine tune the algorithm. Most of these parameters have the same meaning here as they do in the referenced article.

1. C1 is the constant used in the recursive high pass filter that is applied to remove any D.C. bias.

2. C2 is the constant used to vary the weight assigned to the amplitude and first difference in the characteristic function.

3. C3 is the timing constant, used to compute the short term average of the characteristic function.

4. C4 is the timing constant used to compute the long term average of the characteristic function.

5. C5 is the constant used to compute the threshold reference level. A potential event is declared when the short term average becomes larger than C5 times the long term average.

6. C6 is the timing constant used to compute the running mean absolute

value of the filtered data.

7. A station is assumed to be dead when the absolute value of the characteristic function is greater than C7.

8. C8 is used to determine the signal termination level. The signal is terminated when its absolute value falls below this level for D8 seconds. There are currently two different algorithms in use so C8 has two different interpretations. If C8 is positive, then the termination level is C8 times the running mean absolute value of the signal just before the event was declared. This method is useful if the background level at a station is large. If C8 is negative, then the termination level is the absolute value of C8. This will give more consistent terminations from station to station if the noise level is well below this termination level.

9. D5 is the minimum duration in seconds for an event to be declared valid.

10. D9 is the duration in seconds used to initialize the long term average of the characteristic function.

11. I3, I4, and I6 are integer constants used during the validation phase and should not be changed.

HEADER CHANGES: The time of the pick is stored into A; the quality and sense of motion is stored into KA; the end of the event is stored into F.

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

WARNING MESSAGES: 1910: No valid pick found for the following file(s):

SEE COMMANDS: OHPF, OAPF Rex V. Allen, Automatic Earthquake Recognition and Timing from Single Traces, BSSA, Vol. 68, No. 5, Oct. 1978.

LATEST REVISION: May 15, 1987 (Version 10.2)

3.9.2 Cciph

3.9.3 Chpf

SUMMARY: Closes the currently open HYPO pick file.

SYNTAX: CHPF

DESCRIPTION: Automatically appends the instruction card “10” to the end of the file being closed.

SEE COMMANDS: OHPF, WHPF

LATEST REVISION: March 20, 1992 (Version 10.6e)

3.9.4 Ocipf

3.9.5 Ohpf

SUMMARY: Opens a HYPO formatted pick file.

SYNTAX: OHPF {file}

INPUT: file : Name of file to open. If a file by that name already exists, it is opened and new picks are added at the bottom.

DEFAULT VALUES: OHPF HPF

DESCRIPTION: The HYPO pick file generated by SAC can be used as input to HYPO71 and similiar event location programs. Picks from the automatic picker (APK) and manual pick plot (PLOT PK) commands are written into this file once open. This file can be closed using the CHPF command. Opening of a new HYPO pick file automatically closes the previously open one. Opening an existing HYPO pick file automatically deletes the last line of the file, which should be the instruction card “10” that indicates the end of the HYPO input file. Terminating SAC also automatically closes any open pick files. Event delimiters can be written into a HYPO pick file with the WHPF command. See the reference for details on the format of each card.

ERROR MESSAGES: 1901: Can’t open HYPO pick file

- Probably an illegal character in filename. - Occasionally a system error.

If error persists contact the programmer.

SEE COMMANDS: APK, PLOT PK, WHPF, CHPF W.H.K.

Lee and J.C. Lahr; **HYPO71 (Revised):** A Computer Program for

Determining Hypocenter, Magnitude, and First Motion Pattern of Local Earthquakes; U. S. Geological Survey report 75-311.

LATEST REVISION: March 20, 1992 (Version 10.6e)

3.9.6 Whpf

SUMMARY: Writes auxiliary cards into the HYPO pick file.

SYNTAX: WHPF IC n m

INPUT: IC n m : Insert an “instruction card” with the two integers n and m in columns 18 and 19. Allowed values for n are 0, 1, 5, and 6. Allowed values for m are 0, 1, and 9.

DESCRIPTION: The “instruction card” can be used to separate events in a HYPO pick file. See the HYPO71 manual for details on the use of this card. Closing an open HYPO pick file (CHPF command) or quitting SAC automatically appends the “10” instruction card to the HYPO pick file.

ERROR MESSAGES: 1908: HYPO pick file not open.

SEE COMMANDS: CHPF, OHPF W.H.K. Lee and J.C. Lahr; HYPO71 (Revised): A Computer Program for Determining Hypocenter, Magnitude, and First Motion Pattern of Local Earthquakes; U. S. Geological Survey report 75-311.

LATEST REVISION: March 20,1992 (Version 10.6e)

3.10 SCM: Signal Correction Module

3.10.1 Decimate

SUMMARY: Decimates (downsamples) data, including an optional anti-aliasing FIR filter.

SYNTAX: DECIMATE {n},{FILTER {ON—OFF}}

INPUT: n : Set decimation factor to n. Range is 2 to 7. This command may be applied several times if a larger decimation factor is required.

FILTER {ON} : Turn anti-aliasing FIR filter on.

FILTER OFF : Turn anti-aliasing FIR filter off.

DEFAULT VALUES: DECIMATE 2 filter on

DESCRIPTION: This command is used to downsample data after it has been read into memory. An optional finite impulse response (FIR) filter is applied to the data as it is being decimated to prevent aliasing effects normally associated with downsampling digitized analog signals. These filters also preserve the phase information. The application of these FIR filters often produces undesirable transients at each end of the data so the results should be checked graphically. Turning the anti-aliasing filter option off should only be done when the accuracy of the high frequency response is unimportant, such as when plotting.

EXAMPLES: To reduce the sampling rate by a factor of 42:

u: READ FILE1

u: DECIMATE 7

u: DECIMATE 6

HEADER CHANGES: NPTS, DELTA, E, DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1003: Value out of allowed range at symbol
- Range on decimation factor is 2 to 7. **1301:** No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

The decimation by 7 filter has occasionally been unstable.

LATEST REVISION: May 15, 1987 (Version 10.2)

3.10.2 Interpolate

SUMMARY: Interpolates evenly or unevenly spaced data to a new sampling rate.

SYNTAX: INTERPOLATE {DELTA v},{EPSILON v}, {BEGIN v—OFF},{NPTS n—OFF}

INPUT: DELTA v : Set new sampling rate to v.

EPSILON v : Set interpolation convergence factor to v.

BEGIN v : Start interpolation at v. This value becomes the begin time of the interpolated file.

BEGIN OFF : Start interpolation at begin time of uninterpolated file.

NPTS n : Force the number of points in interpolated file to be n.

NPTS OFF : Let SAC calculate the number of points in interpolated file using the begin and end times and new sampling rate.

DEFAULT VALUES: INTERPOLATE DELTA 0.025 EPSILON 0.0001
BEGIN OFF NPTS OFF

DESCRIPTION: This command uses the Wiggins interpolation method (see reference) to convert unevenly spaced data to evenly spaced data and to resample evenly spaced data to a different sampling rate.

EXAMPLES: Assume that FILEA is an unevenly spaced data file. To convert it to an evenly spaced file with a sampling interval of 0.01 seconds:

u: READ FILEA

u: INTERPOLATE DELTA 0.01

Assume that FILEB is an evenly spaced data file with a sampling interval of 0.025. To convert it to a sampling rate of 0.02 seconds:

u: READ FILEB

u: INTERPOLATE DELTA 0.02

Assume that you want to force the interpolated file to have a sampling rate of 0.005, to begin at time 18.237, and to have exactly 400 data points:

u: READ FILEC

u: INTERPOLATE DELTA 0.005 BEGIN 18.237 NPTS 400

WARNING MESSAGES: 2008: Requested begin time is less than files begin time. Output truncated.

- Requested begin time is less than file begin time. Output truncated.

2009: Requested end time is greater than files end time. Output truncated.

- Requested end time is greater than file end time. Output truncated.

Wiggins, 1976, BSSA, 66, p.2077.

HEADER CHANGES: DELTA, NPTS, E, (LEVEN if unevenly spaced.)

LATEST REVISION: July 22, 1991 (Version 10.4c)

3.10.3 Linefit

SUMMARY: Computes the best straight line fit to the data in memory and writes the results to header blackboard variables.

SYNTAX: LINEFIT

DESCRIPTION: A least squares curve fit to a straight line is calculated. The slope, y intercept, standard deviation of the slope, standard deviation of the y intercept, standard deviation of the data and correlation coefficient between the data and the linear fit are written to blackboard variables SLOPE, YINT, SDSLOPE, SDYINT, SDDATA and CORRCOEf respectively. The data does not have to be evenly spaced.

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: none

LATEST REVISION: September 12, 1995 (Version 00.38)

3.10.4 Quantize

SUMMARY: Converts continuous data into its quantized equivalent.

SYNTAX: QUANTIZE [GAINS n ...],[LEVEL v],[MANTISSA n]

INPUT: GAINS n ... : Set list of allowed gains. They must be monotonically decreasing. The maximum number of allowed gains is 8.

LEVEL v : Set the quantization level of the lowest gain. This is the value of the least significant bit in volts.

MANTISSA n : Set the number of bits in the mantissa.

DEFAULT VALUES: QUANTIZE GAINS 128 32 8 1 LEVEL 0.00001
MANTISSA 14

DESCRIPTION: This command exercises a quantization algorithm equivalent to the “rounding” quantization described in Oppenheim and Schaffer (1975, Fig. 9.1). The number of bits used in this algorithm are partitioned into the bits used to represent the characteristic (exponent), the sign bit, and the mantissa bits. The user can specify the number of bits used for the mantissa. The quantization level (value of least significant bit or LSB) can also be specified by the user. The default quantization

level is 10 microvolts. The error of the signal represented by this quantized function is numerically equal to one-half of this quantization level. In the spectral domain, this **error or quantization noise is:** $\text{ERROR} = \frac{1}{12} \Delta \text{LEVEL}^2$ where Δ is the sampling interval. This quantization noise is measured in units of counts*counts/Hz, as a power spectral density. The rms-squared quantization noise is $\frac{1}{6} (\text{LEVEL})^2$. However, this is an accurate approximation to the noise due to quantization only if the rms level of the signal is much larger than the rms quantization noise. In other words, if the signal is not resolved by several hundred counts, then there is a correlation between the quantization noise and the signal being quantized. The fraction of correlation is approximately equal to the ratio of the LEVEL to the rms of the signal being quantized (see Fig. 11.13, Oppenheim and Schaffer, 1975). The gains can be specified by the user to simulate the gain steps in an automatic gain-ranging system. The default gains are those of the Regional Seismic Test Network (RSTN.) Oppenheim, Alan V., and Ronald W. Schaffer; Digital Signal Processing; Prentice-Hall; 1975; 585pp.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: May 15, 1987 (Version 10.2)

3.10.5 Reverse

SUMMARY: Reverse the order of data points.

SYNTAX: REVERSE

DESCRIPTION: This command reverses the order of data points in each file in memory.

LATEST REVISION: May 15, 1987 (Version 10.2)

3.10.6 Rglitches

SUMMARY: Removes glitches and timing marks.

SYNTAX: RGLITCHES options **where options are one or more of the following:**

THRESHOLD *v* TYPE LINEAR—ZERO WINDOW ON—OFF—pdw
METHOD ABSOLUTE—POWER—RUNAVG

INPUT: THRESHOLD *v* : Set onset threshold level to *v*. Data points whose absolute values are greater than or equal to *v* are corrected.

TYPE LINEAR : Correct data points above the threshold by linearly interpolating between the data points on each side of the bad data.

TYPE ZERO : Correct data points above the threshold by setting them to zero.

METHOD ABSOLUTE : Corrects data points having absolute values \geq the threshold *v*.

METHOD POWER : Corrects data points where the power of the signal computed using a backward difference method exceeds the threshold *v*.

METHOD RUNAVG : Corrects data points by calculating a running average and standard deviation in a window SWINLEN seconds long that moves from the end of the trace to the beginning of the trace in 1-point increments. Each new point is compared to the average, and if it differs by more than THRESH2 times the current standard deviation, and if the difference is greater than MINAMP counts, it is replaced by the current mean. This method is always applied to the entire seismogram. **There are three options associated with the RUNAVG method. These are:** **SWINLEN *v* :** Set length in seconds of running average window. **THRESH2 *v* :** Set the threshold value for glitches. **MINAMP *v* :** Set the minimum amplitude for glitches.

WINDOW ON : Only correct data points within the previously defined pdw.

WINDOW OFF : Correct data points within the entire data file.

WINDOW pdw : Only correct data points within the defined pdw. A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to make measurements on. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

DEFAULT VALUES: RGLITCHES THRESHOLD 1.0E+10 TYPE

LINEAR WINDOW OFF METHOD ABSOLUTE SWINLEN 0.5 THRESH2
5.0 MINAMP 50

DESCRIPTION: This command can be used to smooth out irregularities caused by “glitches” in the data acquisition system and by timing marks produced by some data acquisition systems. It checks each data point to see if it’s value is greater than or equal to the requested “onset threshold level”. It then zeros out these bad data points or linearly interpolates between the data point just before and the data point just after the bad ones. You can have it remove glitches in the entire file or select a smaller portion of the file by setting the window. Using this option lets you remove glitches that are smaller than the maximum in the entire data file.

EXAMPLES: Some examples of pdw are given below:

B 0 30: First 30 secs of the file.

T3 -1 T7: From 1 sec before T3 time pick to T7 time pick.

30.2 48: 30.2 to 48 secs relative to file zero.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

1307: Illegal operation on spectral file

LATEST REVISION: March, 1997 (Version 00.53a)

3.10.7 Rir

3.10.8 Rmean

SUMMARY: Removes the mean.

SYNTAX: RMEAN

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMEN

LATEST REVISION: October 11, 1984 (Version 9.1)

3.10.9 Rotate

SUMMARY: Rotates a pair of data components through an angle.

SYNTAX: ROTATE {to GCP—TO v—THROUGH v—,{NORMAL—REVERSED}}

INPUT: TO GCP : Rotate to the “great circle path”. Both components must be horizontals. The station and event coordinates header fields must be defined.

TO v : Rotate to the angle v in degrees. Both components must be horizontals.

THROUGH v : Rotate through the angle v in degrees. One component may be vertical.

NORMAL : Output components with normal polarity.

REVERSED : Output components with reversed polarity.

DEFAULT VALUES: ROTATE TO GCP NORMAL

DESCRIPTION: Pairs of data components are rotated in this command. Each pair must have the same station name, event name, and sampling rate. In the THROUGH option both components are simply rotated through the requested angle. One of those components may be a vertical. Rotations in the horizontal plane are clockwise from north. Rotations with a vertical component are clockwise from up. Both components must be horizontals when the TO option is used. This means that CMPAZ must be defined and that CMPINC must be 90 degrees. After the rotation is completed the first component of each pair will be directed along the angle given after to TO keyword. If the TO GCP option is used this component will be directed along the angle given by the station-event back azimuth plus or minus 180 degrees. This component therefore points from the event toward the station. The station and event coordinates header fields (STLA, STLO, EVLA, and EVLO) must be defined so that the back azimuth can be calculated. The NORMAL and REVERSED options also apply only to horizontal rotations. If the NORMAL option is used the second component leads the first by 90 degrees. If the REVERSED option is used it lags the first by 90 degrees.

EXAMPLES: To rotate a pair of horizontals through 123.43

degrees:

u: READ XYZ.N XYZ.E

u: ROTATE TO 123.43

To rotate two sets of horizontals to the great circle path:

u: READ ABC.N ABC.E DEF.N DEF.E

u: ROTATE TO GCP

u: W ABC.R ABC.T DEF.R DEF.T

In the above example if the BAZ header variable had been 33 degrees, the radial components would be at 213 degrees and the tangential components at 303 degrees. If reversed polarity had been requested the tangential components would be at 123 degrees.

HEADER CHANGES: CMPAZ, CMPINC

ERROR MESSAGES: 1301: No data files read in.

2001: Command requires an even number of data files.

2004: Insufficient header information for rotation:

- STLA, STLO, EVLA, EVLO must be defined for GCP option. **2002:**

Following files are not an orthogonal pair:

2003: Following files are not both horizontals:

- TO option only works on horizontals.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.10.10 Rq

SUMMARY: Removes the seismic Q factor from spectral data.

SYNTAX: RQ [Q v],[R v],[C v]

INPUT: Q v : Set quality factor to v.

R v : Set distance in km. to v.

C v : Set group velocity in km/sec to v

DEFAULT VALUES: RQ Q 1. R 0. C 1.

DESCRIPTION: The equation used to correct the amplitude is given below:

$$\text{AMP_corrected}(F) = \text{AMP_uncorrected}(F) * \text{Exp}((\pi * R * F) / (Q * C))$$

where: F is the frequency in Hz. R is the distance in km. C is the group velocity in km/sec. Q is the the nondimensional quality factor.

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES: 1301: No data files read in.

1305: Illegal operation on time series file

WARNING MESSAGES: 1604: Following file now in amplitude-phase format:

- file was in real-imaginary format.

LIMITATIONS: Can only handle constants for the various parameters. Modifications to allow these parameters to vary with frequency may be added at a later date.

LATEST REVISION: January 8, 1983 (Version 8.0)

3.10.11 Rtrend

SUMMARY: Removes the linear trend.

SYNTAX: RTREND

DESCRIPTION: A least squares curve fit to a straight line is calculated. This straight line or trend is then “subtracted” from the data. The data does not have to be evenly spaced.

OUTPUT: The best fitting straight line parameters for the last file in the data file list are written to blackboard variables beginning with RTR. RTR_SLP is the slope of the line. RTR_SDSLPL is the standard deviation in the slope. RTR_YINT is the y intercept of the line. RTR_SDYINT is the standard deviation in the y intercept. RTR_SDDTA is the standard deviation in the data. RTR_CORRCF is the data correlation coefficient.

ERROR MESSAGES: 1301: No data files read in.

1307: Illegal operation on spectral file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMIN

LATEST REVISION: February 11, 2000 (Version 58)

3.10.12 Smooth

SUMMARY: Applies an arithmetic smoothing algorithm to the data.

SYNTAX: SMOOTH {MEAN—MEDIAN},{HALFWIDTH n}

INPUT: MEAN : Apply a mean (average) smoothing algorithm.

MEDIAN : Apply a median point smoothing algorithm.

HALFWIDTH n : Set halfwidth of smoothing window to n. The moving window will contain n points on each side of the point being smoothed.

DEFAULT VALUES: SMOOTH MEAN HALFWIDTH 1

DESCRIPTION: This command applies an arithmetic smoothing algorithm to each data point. The type of algorithm and the size of the sliding window around each data point can be varied. The size of the window is defined by specifying its halfwidth. This forces the moving window to be centered around each data point and forces the window size to be an odd number of points, which makes the algorithms easier and less ambiguous.

HEADER CHANGES: DEPMIN, DEPMAX,DEPMEN

LATEST REVISION: April 13, 1987 (Version 10.1)

3.10.13 Stretch

SUMMARY: Stretches (upsamples) data, including an optional interpolating FIR filter.

SYNTAX: STRETCH {n},{FILTER {ON—OFF}}

INPUT: n : Set upsampling factor. Must be in the range 2 to 7.

FILTER {ON} : Turn interpolating filter option on.

FILTER OFF : Turn interpolating filter option off.

DEFAULT VALUES: STRETCH 2 FILTER ON

DESCRIPTION: By using the interpolating filter option, this command can be used to create a file with a smaller sampling interval (more data points) but which looks similar to the original. Care should be taken when using this command, because the filter does effect the frequency content. When this filter option is off, the appropriate number of zeros are simply inserted between each of the original data points.

HEADER CHANGES: NPTS, DELTA, E, DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: May 15, 1987 (Version 10.2)

3.10.14 Taper

SUMMARY: Applies a symmetric taper to each end of data.

SYNTAX: TAPER {TYPE HANNING—HAMMING—COSINE},{WIDTH
v}

INPUT: TYPE HANNING : Apply a Hanning taper.

TYPE HAMMING : Apply a Hamming taper.

TYPE COSINE : Apply a cosine taper.

WIDTH v : Set the taper width on each end to v. This is a value between 0.0 and 0.5.

DEFAULT VALUES: TAPER TYPE HANNING WIDTH 0.05

DESCRIPTION: A taper is a monotonically varying function between zero and one. It is applied in a symmetric manner to the data such that the signal is zero for the first and last data points and increases smoothly to its original value at an interior point relative to each end. **The general form for the taper is:**

$$\text{DATA}(J)=\text{DATA}(J)*(F0-F1*\text{COS}(\text{OMEGA}*(J-1)))$$

This equation would be applied to the left hand side of each signal. A symmetric one is applied to the right hand side. The following table defines the various parameters used in the different tapers. In this table N is the length of the taper on each end.

TYPE	OMEGA	F0	F1
HANNING	PI/N	0.50	0.50
HAMMING	PI/N	0.54	0.46
COSINE	$\text{PI}/(2*N)$	1.00	1.00

The figure on the next page shows a blowup of the left hand side of each taper to give you a better idea of their shapes.

ERROR MESSAGES: 1301: No data files read in.

1306: Illegal operation on unevenly spaced file

HEADER CHANGES: DEPMIN, DEPMAX, DEPMEN

LATEST REVISION: January 15, 1985 (Version 9.10) A Comparison of the Various Types of Tapers

3.11 SPE: Spectral Estimation Subprocess

3.11.1 Cor

3.11.2 Mem

3.11.3 Mlm

3.11.4 Pds

3.11.5 Plotcor

3.11.6 Plotpe

3.11.7 Plotspe

3.11.8 Prewhiten

3.11.9 Quitsub

SUMMARY: Terminates the currently active subprocess.

SYNTAX: QUITSUB

DESCRIPTION: This command terminates the currently active subprocess, returning to the main SAC command environment. Files in memory are retained. There are **currently two subprocesses available in SAC:**

SES Spectral Estimation Subprocess

SSS Signal Stacking Subprocess

SEE COMMANDS: SES, SSS

LATEST REVISION: October 11, 1984 (Version 9.1)

3.11.10 Read

SUMMARY: Reads data from SAC data files on disk into memory.

SYNTAX: READ [options] [filelist] **where options is one or more of the following:**

MORE TRUST ON—OFF COMMIT—ROLLBACK—RECALLTRACE
DIR CURRENT—name XDR ALPHA SEG Y SCALE [ON—OFF]

ALL options MUST precede any element in the filelist.

INPUT: MORE : Place the new data files in memory AFTER the old ones. If this option is omitted, the new data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

TRUST ON—OFF : This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

COMMIT : If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory – removing any previous versions of headers or waveforms from RAM – prior to reading more files. COMMIT is the default.

ROLLBACK : If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE : If the MORE option is specified, the RECALLTRACE option: - reverts to the last committed version of the waveform, - reverts to the last committed version of those header variables closely linked to the waveform, - commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of header variables which are committed, and which are rolled back.)

DIR CURRENT : Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name : Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

XDR : The input files are in XDR format. This format is used for moving binary data files to/from a different architecture, such as a pc running LINUX.

ALPHA : The input files are SAC formatted alphanumeric (ascii) files. the ALPHA option is incompatible with the XDR option.

SEGY : Read file formatted according to the IRIS/PASSCAL form of the SEG Y format. This format allows one waveform per file.

SCALE : Used only in conjunction with the SEG Y option, the SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the SEG Y files. When SCALE is ON, SAC multiplies counts by a SCALE factor given in the file. This scale option changes with the SCALE options of READCSS, READGSE, READDB, and READSUDS. If SCALE is OFF, the SCALE value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The preferred method is with the TRANSFER command. It is recommended to leave SCALE off for the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

filelist : file—wild ...

file : A legal filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one. See the DESCRIPTION and EXAMPLES sections below for more details.

wild : A wildcard laden token that expands to a list of filenames. See the DESCRIPTION and EXAMPLES sections below and the WILD command for more details.

DEFAULT VALUES: READ COMMIT DIR CURRENT

DESCRIPTION: All commands in SAC work on the data that is currently in memory. This data in memory is analogous to the temporary or working files used by a text editor. The READ command transfers data from one or more disk files into memory. The default is to read all of the data from each disk file. The CUT command can be used to specify that only a portion of each disk file be read. SAC files produced in or after the year 2000 are presumed to have a four digit value for the year. Files with two digit year values will be assumed to be in the twentieth century, and will be incremented by 1900. Normally all data in memory prior to the execution of another READ command is lost. The new data replaces the old data. If the keyword MORE is the second symbol in the command, the new data is placed in memory after the old data. The data file list becomes the concatenation of the old file list and the new file list. **There are three cases where the MORE option may be useful:**

- (1) The filelist is too long to be typed on one line.
- (2) A name was misspelled in a long filelist.
- (3) A file is read, some analysis performed, and a comparison with the original is desired.

Examples of each of these cases are given below. The filenames may be simple filenames in the current directory or they may be absolute or relative pathnames pointing to other directories on your **system**. **Examples of absolute pathnames are:**

UNIX: /dir/subdir/file

VMS: disk:[dir.subdir]file

PRIMOS: ;disk;dir;subdir;file

Examples of relative pathnames are:

UNIX: subdir/file

PRIMOS: *;subdir;file

In the above examples “disk” is the name of a physical disk partition, “dir” is the name of a top level directory, “subdir” is a subdirectory of that partition, and “file” is a file in that subdirectory. In general there is no limit on the nesting of subdirectories. Filenames may also contain wild-card characters. You can use them match a single character, to match zero or

more characters, and to form groupings of characters. Some examples are given below. See the WILD command for more examples and a complete explanation of all the wildcarding options.

*** Important *** SAC has two data buffers; this is what allows SAC to provide the COMMIT, ROLLBACK and RECALLTRACE commands. One data buffer stores the header information in SAC format, and the second stores headers in CSS 3.0 format. This CSS 3.0 data buffer allows seamless consistency with CSS 3.0 in READCSS and WRITECSS; it also allows direct access to the CSS 3.0 formatted Oracle database. In CSS (a relational format), it is important to maintain consistency with the event IDs (evid, or nevid in SAC). In SAC format (a very flat format), such consistency is not as important, and in some cases, it is lost. Anytime data is loaded into SAC, it is stored in both buffers. When transferring data from SAC to CSS data buffers, there is a potential ambiguity in handling event information. If matching evids are found, it could be that the two files have identical event information, or it could be that the match is an artifact of the merge of these two different data formats within SAC. Two peices of information are involved in resolving this ambiguity, one is the history of data loaded into SAC memory, and the other is the confidence the user sets with the TRUST ON—OFF option on the command line of most Read commands and ADDSTACK. It is expected that the user will have some idea if the data files are consistent, if they share event information, etc. The history of data loaded into SAC memory begins when data is loaded into memory without the MORE option, and ends the next time data is loaded into memory without the MORE option. Any time in between that data is loaded into memory with the MORE option, it becomes part of the existing history. All commands which load data into memory are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. The READDDB command is the most reliable way to load data into SAC because the database insures consistency. For this reason, the levels of confidence (in ascending order) are LOW, HIGH, and RDB. TRUST is an option set on the commandline of most commands that load SAC data. TRUST can be ON or OFF. Each time a command loads

data into SAC, it responds to both the confidence level and the TRUST to determine how to treat matching event IDs. When requisite levels of TRUST and confidence are present, matching event IDs are treated as an indicator that the two files share identical event information. This being the case, the event IDs are left unchanged. When requisite levels of TRUST and confidence are not met, matching event IDs are **are treated as artifacts of the merging of two different data formats:** SAC and CSS. READDB, being the only truly reliable way to load data, will always treat the data as reliable as long as READDB is the only method used to load data into SAC. In this case, the TRUST will not be used, and the confidence lever will always be set to RDB. If any other data loading method is used, then the confidence will be reduced to HIGH, or LOW, and RDB will respond similarly to the next set of commands. **The following commands are considered HIGH confidence:** READ, READCSS, READHDR, and ADDSTACK. These commands will consider both the confidence level and the TRUST in determining how to handle event ID matches. If the confidence is HIGH (or RDB) and the TRUST is ON, then confidence is set to HIGH, and the event IDs are treated as reliable. However if the TRUST is OFF or if the confidence level is LOW, then the confidence is set to LOW and the matching event IDs are treated as artifacts, and new IDs are generated for the incoming data file. The following commands are considered LOW confidence because event ID **information is not available:** READTABLE, READGSE, READSUDS, FUNCGEN, DATAGEN, READSP, READSDD, and READ with the SEG Y option and READCSS when the input files are in the CSS 2.8 data format. These commands will always generate event IDs and set the confidence level to LOW. Commands written by the user and added to SAC via the external command interface (the LOAD command) are a special case, since the user writes the code. In these cases, the confidence is based entirely on the TRUST that the user sets. Hence, it is incumbent upon the user to set the TRUST either on the commandline or within the code. For more information use HELP EXTERNAL. Within a given history, if data is loaded into SAC by any means other than READDB, the data is probably not consistent with the database, and should probably not be loaded back into the database,

unless the user takes responsibility to insure consistency among events.

EXAMPLES: In the following examples is it assumed that the following SAC data **files are in your current disk directory:** F01, F02, F03, and G03. In these examples, the UNIX wildcard characters (e.g., “?” matches any single character and “*” matches zero or more characters) are used. See the WILD command for more information on how to use wildcards. To read the first three **files:**

u: READ F01 F02 F03

The following command produces the same result using the wildcard operator:

u: READ F*

This command also produces the same result by using the concatenation operator:

u: READ F0[1,2,3]

To read the second, third, and fourth files:

u: R F02 ?03

The following examples show the use of the MORE option:

u: R F03 G03

... files F03 and G03 are in memory.

u: R F01 F02

... files F01 and F02 are in memory.

u: R MORE F03 G03

... files F01, F02, F03, and G03 are in memory.

This example uses the MORE option when a filename was misspelled:

u: R F01 G02 F03

s: WARNING: File does not exist: G02

s: Will read the remainder of the data files.

... files F01 and F03 are in memory.

u: R MORE F02

... files F01, F03, and F02 are now in memory.

... note the order of the files in this case.

If you wanted to apply a highpass filter to a data file and then graphically compare the results to the original:

```
u: READ F01
u: HIGHPASS CORNER 1.3 NPOLES 6
u: READ MORE F01
u: PLOT1
```

... plot shows filtered and original data

Now assume you were in the directory “/me/data” when you started up SAC and that you wanted to work with the data files in the subdirectories “event1” and “event2”:

```
u: READ DIR EVENT1 F01 F02
... files in directory /me/data/event1 are read.
u: READ F03 G03
```

... files in same directory are read.

```
u: READ DIR EVENT2 *
```

... all files in /me/data/event2 are read.

```
u: READ DIR CURRENT F03 G03
```

... files in directory /me/data are read.

Note: For examples of the differing behavior between the COMMIT, ROLLBACK, RECALLTRACE options, see the commands of the same names.

ERROR MESSAGES: 1301: No data files read in.

- haven’t given a list of files to read. - none of the files in the list could be read. **1320:** Available memory too small to read file

1314: Data file list can’t begin with a number.

1315: Maximum number of files in data file list is

6002: No more data-sets available.

WARNING MESSAGES: 0101: opening file

0108: File does not exist:

0114: reading file

- Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the READERR command.

HEADER CHANGES: E, DEPMIN, DEPMAX, DEPMEN, B if cut option is on.

SEE COMMANDS: CUT, READERR, WILD, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION: June. 18, 1999 (Version 0.58)

3.11.11 Readcor

3.11.12 Spe

Introduction

Introduction SPE is a spectrum estimation package intended primarily for use with stationary random processes. **It contains three different spectral estimation techniques:** Power Density Spectra, Maximum Likelihood Method, and Maximum Entropy Method. These are all indirect methods, because they use a sample correlation function, rather than the data itself, to estimate the spectral content.

This is a SAC subprocess. A subprocess is like a small program within the main SAC program. You start a subprocess by typing its name (SPE in this case.) You can terminate it and return to the main program using the QUITSUB command. You can also terminate SAC from within a subprocess using the QUIT command. While within a subprocess, you can execute any command belonging to that subprocess plus a limited number of main SAC commands.

SPE Commands COR Computes the correlation function.

MEM Calculates the spectral estimate using Maximum Entropy Method.

MLM Calculates the spectral estimate using Maximum Likelihood Method.

PDS Calculates the spectral estimate using Power Density Spectra Method.

PLOTCOR Plots the correlation function.

PLOTPE Plots the RMS prediction error function.

PLOTSPE Plots the spectral estimate.

QUITSUB Terminates a SAC subprocess.

READ Reads data from a SAC data file into memory.

WRITECOR Writes a SAC file containing the correlation function.

WRITESPE Writes a SAC file containing the spectral estimate.

Their abbreviated names are also allowed.

Main SAC Commands executable from within the SPE subprocess AXES
BEGINDEVICES BEGINFRAME BEGINWINDOW BORDER COLOR
COMCOR COPYHDR DATAGEN ECHO ENDDEVICES ENDFRAME
ERASE EVALUATE FLOOR GETBB GRID GTEXT HELP INSTALL-
MACRO LINE LINLIN LINLOG LOGLAB LOGLIN LOGLOG MACRO
MESSAGE PAUSE PLABEL PLOTQ QDP QUIT READALPHA READBBF
REPORT SETBB SETDATADIR SETDEVICE SETMACRO SGF SYM-
BOL SYNTAX SYSTEMCOMMAND TICKS TITLE TSIZE VSPACE WAIT
WINDOW WRITEBBF XDIV XFUDGE XFULL XGRID XLABEL XLIM
XLIN XLOG XVPORT YDIV YFUDGE YFULL YGRID YLABEL YLIM
YLIN YLOG YVPORT

The Theory

Overview SPE is a spectrum estimation package intended primarily for use with stationary random processes. It implements three different indirect spectral estimators. They are called indirect, because they do not estimate the spectrum directly from the data, but from a sample correlation function that is computed from the data. The choice of indirect methods is a matter of taste, since direct spectral estimation techniques are also available. The correlation function itself is a useful quantity. You may wish to examine it in the course of performing spectral estimation tasks. The choice of indirect techniques is supported by “Spectral Analysis and Its Application,” by Jenkins and Watts, a respected reference on the subject of spectrum estimation. The type of spectrum estimated by SPE is properly described as the power density spectrum, with the spectrum defined in the frequency domain. Thus, the estimated power delivered by the random process in some band of frequencies is the integral of the spectral power density estimate over that band of frequencies.

User Control SPE affords the user some control over the details of estimation process. For some, with experience in estimating spectra, this is highly desirable. Defaults are provided for those who do not wish to become

involved in the details of the theory. The user has a choice of data window type, size, and the number of windows used when estimating the correlation function. Generally these parameters control the resolution of the estimate, and the amount of reduction of variance desired in the final estimate. In addition, prewhitening of the data may be specified as part of the process of estimating the correlation function. Prewhitening often has the effect of mitigating a severe “window bias” that can occur in spectral estimates having a high dynamic range. The warping of the spectrum that occurs with prewhitening is compensated for in the final result. In this implementation, low-order prediction error filters are used for prewhitening.

The Estimators **The user has a choice of three spectral estimators:** Power Density Spectra (PDS), Maximum Likelihood Method (MLM), and Maximum Entropy Method (MEM).

PDS **The PDS estimator is quite simple:** the sample correlation function is multiplied by a correlation window, then the result is transformed with an FFT to obtain the spectral estimate. The user again has a choice of the window type and the size of the window. The above mentioned book by Jenkins and Watts could be considered as the detailed documentation for the PDS technique.

MLM The MLM estimator generates a spectral estimate which is the power output of a bank of narrow band-pass filters which have been optimized to reject out-of-band power. The result is a smoothed, parametric estimate of the power density spectrum. The user can choose the number of parameters. Documentation for this method can be found in the paper by Richard Lacoss in the IEEE book “Modern Spectrum Analysis” by Donald Childers.

MEM The MEM estimator is another parametric method, which uses a prediction error filter to whiten the data. The resulting spectral estimate is proportional to the inverse of the filter’s power frequency response. The user is free to choose the order of the prediction error filter. Documentation for this method can be found in the review paper on linear prediction by John Makhoul in “Modern Spectrum Analysis.” The formal name of the actual method implemented is the Yule-Walker method.

Diagnostics In addition to the spectrum, several diagnostic functions can be calculated and plotted. The prediction error can be plotted as a function of order. This plot can be used to select a good size for the prediction error filter used in the MEM method. Since much is known about the performance of the PDS estimator, more diagnostic information is available for this method in SPE. The 90% confidence limits can be estimated theoretically, as can the frequency resolution of the estimate. Both of these quantities can be indicated on a PDS spectral plot.

The Program Differences There are two primary differences between SPE and the main SAC program. Only one data file can be processed by SPE at a time. This is because SPE produces and stores a number of auxiliary functions (the correlation function, the prediction error function, and the spectral estimate itself) as it proceeds. This restriction to a single data file may be removed in the future. The second difference is that, unlike SAC itself, there is a specific order or progression in which the commands are generally executed.

Initialization This progression begins when the SPE command is executed. Default values for the various SPE parameters are defined at this time. The data file may have been read in using the READ command before entering SPE or at any time while within SPE. Space for the above mentioned auxiliary functions is created whenever a new file is read.

Correlation The correlation function is then computed, using the COR command. It may be saved as a SAC data file using the WRITECOR command and later read back into SPE using the READCOR command. This is more efficient than recomputing the correlation each time, especially if the data file is very long. At this point, you may wish to examine the correlation function using the PLOT COR command. You may also wish to examine the prediction error function using the PLOT PT command if you are going to use the MEM method.

Estimation Now you are ready to select one of the three spectral estimation techniques using the PDS, MLM, or MEM commands. Each technique has its own options. You may now examine the resulting spectrum using the PLOT SPE command. There are several different scaling options avail-

able. You can also save the spectral estimate as a SAC data file using the WRITESPE command.

Termination **At this point you have several options:** you can select a different spectral estimate technique, read in a different correlation function, read in a different data file, terminate the subprocess using the QUITSUB command, or terminate SAC using the QUIT command.

3.11.13 Speid

3.11.14 Writecor

3.11.15 Writespe

3.12 SSS: Signal Stacking Subprocess

3.12.1 Addstack

3.12.2 Changestack

3.12.3 Deletestack

3.12.4 Deltacheck

3.12.5 Distanceaxis

3.12.6 Distancewindow

3.12.7 Globalstack

3.12.8 Incrementstack

3.12.9 Liststack

3.12.10 Phase

3.12.11 Plotrecordsection

3.12.12 Plotstack

3.12.13 Quitsub

SUMMARY: Terminates the currently active subprocess.

SYNTAX: QUITSUB

DESCRIPTION: This command terminates the currently active subprocess, returning to the main SAC command environment. Files in memory are retained. There are **currently two subprocesses available in SAC:**

SES Spectral Estimation Subprocess

SSS Signal Stacking Subprocess

SEE COMMANDS: SES, SSS

LATEST REVISION: October 11, 1984 (Version 9.1)

3.12.14 Sss

Introduction SSS is primarily a package for doing signal stacking (i.e. summation or beamforming). Each signal (i.e. SAC file) has properties such as a static delay, epicentral distance, weighting factor, and polarity associated with it. The dynamic delays can be calculated using a normal moveout or refracted wave velocity model. Certain delay properties can be automatically incremented between summations. Files are easily added to or removed from the stack file list. The time window for the stack is easily adjusted. Files which do not contain data throughout the stack time window are filled with zeros. The stack file list can be plotted with or without the summation. Each summation can be saved on disk for later use. A record section plot is also included in this subprocess.

This is a SAC subprocess. A subprocess is like a small program within the main SAC program. You start a subprocess by typing its name (SSS in this case.) You can terminate it and return to the main program using the quitsub command. You can also terminate SAC from within a subprocess using the QUIT command. While within a subprocess, you can execute any command belonging to that subprocess plus a limited number of main SAC commands. A list of the allowed main SAC commands is included in this document.

SAC Signal Stacking Subprocess Manual

The Commands

Overview The remainder of this manual contains the documentation on each of the SSS commands. They are in alphabetical order and are also summarized below. The format and notation is the same as the A list of the allowed main SAC commands is also shown below. You can also use all of the SAC macro features in this subprocess.

SSS Commands

ADDSTACK Add a new file to the stack file list. **CHANGESTACK** Change properties of files currently in the stack file list. **DELETESTACK** Deletes one or more files from the stack file list. **DELTACHECK** Change the sampling rate checking option. **DISTANCEAXIS** Define the record section plot distance axis parameters. **DISTANCEWINDOW** Controls distance window properties on subsequent record section plots. **GLOBALSTACK** Sets global stack properties. **INCREMENTSTACK** Increments properties for files in the stack file list. **LISTSTACK** Lists the properties of the files in the stack file list. **PLOTRECORDSECTION** Plots a record section of the files in the stack file list. **PLOTSTACK** Plots the files in the stack file list. **QUITSUB** Terminates the Signal Stacking Subprocess. **SUMSTACK** Sums the files in the stack file list. **TIMEAXIS** Controls the time axis properties on subsequent record section plots. **TIMEWINDOW** Sets the time window limits for subsequent stack summation. **VELOCITYMODEL** Sets stack velocity model parameters for computing dynamic delays. **VELOCITY-ROSET** Controls placement of a velocity roset on subsequent record section plots. **WRITESTACK** Writes a stack summation to disk. **ZEROSTACK**

Zeros or reinitializes the signal stack.

Main SAC Commands This is a list of the allowed main SAC commands.

Their abbreviated names are also allowed.

AXES BEGINDEVICES BEGINFRAME BEGINWINDOW BORDER
COLOR COMCOR COPYHDR DATAGEN ECHO ENDDEVICES END-
FRAME ERASE EVALUATE FLOOR GETBB GRID GTEXT HELP IN-
STALLMACRO LISTHDR LINE LINLIN LINLOG LOGLAB LOGLIN LOGLOG
MACRO MESSAGE PAUSE PLABEL PLOTQ QDP QUIT READBBF
REPORT SETBB SETDATADIR SETDEVICE SETMACRO SGF SYM-
BOL SYNTAX SYSTEMCOMMAND TICKS TITLE TSIZE VSPACE WAIT
WINDOW WRITEBBF XDIV XFUDGE XFULL XGRID XLABEL XLIM
XLIN XLOG XVPORT YDIV YFUDGE YFULL YGRID YLABEL YLIM
YLIN YLOG YVPORT

3.12.15 Sumstack

3.12.16 Timeaxis

3.12.17 Timewindow

3.12.18 Travelttime

3.12.19 Velocitymodel

3.12.20 Velocityrosette

3.12.21 Writestack

3.12.22 Zerostack

3.13 SMM: Signal Measurement Module

3.13.1 Markptp

SUMMARY: Measures and marks the maximum peak to peak amplitude of each signal within the measurement time window.

SYNTAX: MARKPTP {LENGTH v},{TO marker}

INPUT: LENGTH v : Change the length of the sliding window to v seconds.

TO marker : Define the first time marker in the header to store results. The time of the minimum is stored in this marker. The time of the maximum is stored in the next marker.

marker : T0—T1—T2—T3—T4—T5—T6—T7—T8—T9

DEFAULT VALUES: MARKPTP LENGTH 5.0 TO T0

DESCRIPTION: This command measures the times and the amplitude of the maximum peak-to-peak excursion of the data within the current measurement time window (see MTW.) The results are written into the header. The time of the minimum value (valley) is written into the requested marker. The time of the maximum value (peak) is written into the next marker. The peak-to-peak amplitude is written into USER0. The results are also written into the alphanumeric pick file if it is open (see OAFP.)

EXAMPLES: To set the measurement time window to be between the two header fields, T4 and T5, and the default sliding window length and marker:

u: MTW T4 T5

u: MARKPTP

To set the measurement time window to be the 30 seconds immediately after the first arrival, and the sliding window length to 3 seconds, and the starting marker to T7

u: MTW A 0 30

u: MARKP L 3. TO T7

HEADER CHANGES: T_n, USER0, KT_n, KUSER0

SEE COMMANDS: MTW, OAPF

LATEST REVISION: May 15, 1987 (Version 10.2)

3.13.2 Marktimes

SUMMARY: Marks files with travel times from a velocity set.

SYNTAX: MARKTIMES {TO marker},{DISTANCE HEADER—v},
{ORIGIN HEADER—v—GMT time},{VELOCITIES v ...}

INPUT: TO marker : Define the first time marker in the header to store results. The time markers are incremented for each requested velocity.

marker : T0—T1—T2—T3—T4—T5—T6—T7—T8—T9

DISTANCE HEADER : Use the distance (DIST) from the header in the travel time calculations.

DISTANCE v : Use v as the distance in the travel time calculations.

ORIGIN HEADER : Use the origin time (O) in the header in the travel time calculations.

ORIGIN v : Use v as the offset origin time.

ORIGIN GMT time : Use the Greenwich mean time as the origin time.

time : Greenwich mean time in the form of six integers: year, julian day, hour, minute, second, and millisecond.

VELOCITIES v ... : Set the velocity set to use in the travel time calculations. Up to 10 velocities may be entered.

ALTERNATE FORMS: UTC for Universal Time Coordinate may be used instead of GMT.

DEFAULT VALUES: MARKTIMES VELOCITIES 2. 3. 4. 5. 6.
DISTANCE HEADER ORIGIN HEADER TO T0

DESCRIPTION: This command marks travel times in the header, given the origin time of the event, the epicentral distance, and an input velocity set. The following simple equation is used to estimate travel times.

$$\text{time}(j) = \text{origin} + \text{distance}/\text{velocity}(j)$$

The results are written into the header in the requested time marker.

EXAMPLES: To use the default velocity set but force the distance to be 340 kilometers **and the first marker to be T4:**

u: MARKTIMES DISTANCE 340. TO T4

To select a different velocity set:

u: MARKT V 3.5 4.0 4.5 5.0 5.5

To set the origin time in GMT and store the results in T2:

u: MARKT ORIGIN GMT 1984 231 12 43 17 237 TO T2

HEADER CHANGES: T_n, KT_n

LATEST REVISION: May 15, 1987 (Version 10.2)

3.13.3 Markvalue

SUMMARY: Searches for and marks values in a data file.

SYNTAX: MARKVALUE {GE v—LE v},{TO marker}

INPUT: GE v : Search for and mark the first data point that is greater than or equal to v.

LE v : Search for and mark the first data point that is less than or equal to v.

TO marker : Define the time marker in the header in which to store the result.

marker : T0—T1—T2—T3—T4—T5—T6—T7—T8—T9

DEFAULT VALUES: MARKVALUE GE 1 TO T0

DESCRIPTION: This command searches for the requested value in each data file and marks the time of the first occurrence of that value. If a measurement time window has been defined (see MTW), only that portion of each data file is searched. Otherwise the entire file is searched. The results are written into the header in the requested time marker.

EXAMPLES: To search for the first data point whose value is at least 3.4 and to store the result in the header as T7

u: MARKVALUE GE 3.4 TO T7

To later perform that same search in the measurement time window 10 seconds **long beginning at T4:**

u: MTW T4 0 10

u: MARKVALUE

HEADER CHANGES: T_n, KT_n

SEE COMMANDS: MTW

LATEST REVISION: May 15, 1987 (Version 10.2)

3.13.4 Mtw

SUMMARY: Determines the measurement time window for use in subsequent measurement commands.

SYNTAX: MTW {ON—OFF—pdw}

INPUT: {ON} : Turn measurement time window option on but don't change window values.

OFF : Turn measurement time window off. Measurements are done on the entire file.

pdw : Turn measurement time window on and set window values to a new "partial data window." A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to make measurements on. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

DEFAULT VALUES: MTW OFF

DESCRIPTION: When this option is on, measurements are made on the data within the window only. When this option is off, measurements are made on the entire file. This option currently applies to the MARKPTP and MARKVALUE commands only. Others measurement commands will be added as needed.

EXAMPLES: Some examples of pdw are given below:

B 0 30: First 30 secs of the file.

A -10 30: From 10 secs before to 30 secs after first arrival.

T3 -1 T7: From 1 sec before T3 time pick to T7 time pick.

B N 2048: First 2048 points of file.

30.2 48: 30.2 to 48 secs relative to file zero.

SEE COMMANDS: CUT, MARKPTP, MARKVALUE

LATEST REVISION: May 15, 1987 (Version 10.2)

3.13.5 Rms

SUMMARY: Computes the root mean square of the data within the measurement time window.

SYNTAX: RMS {NOISE ON—OFF—pdw},{TO USERn}

INPUT: NOISE ON : Turn noise normalization option on.

NOISE OFF : Turn noise normalization option off.

NOISE pdw : Turn noise normalization option on and change noise “partial data window.” A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to make measurements on. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

TO USERn : Define the user header variable in which to store the result. n is an integer in the range 0 to 9.

DEFAULT VALUES: RMS NOISE OFF TO USER0

DESCRIPTION: This command computes the root mean square of the data within the current measurement time window (see MTW.) The result is written into one of the floating point user header variables. The result may be corrected for noise **if desired by defining a noise window.** **The general form of the calculation is:** where the first summation is over the signal window and the second is over the optional noise window.

EXAMPLES: To compute the uncorrected root mean square of data between the two header fields, **T1** and **T2**, and to store the result into the **USER4** header field:

u: MTW T1 T2

u: RMS TO USER4

To compute the corrected root mean square using a noise window 5 seconds long **ending at the header field T3:**

u: MTW T1 T2

u: RMS NOISE T3 -5.0 0.0

HEADER CHANGES: USERn

SEE COMMANDS: MTW, CUT

LATEST REVISION: March 22, 1991 (Version 10.6d)

3.14 ICM: Instrument Correction Module

3.14.1 Prewhiten

3.14.2 Transfer

SUMMARY: Performs deconvolution to remove an instrument response and convolution to apply another instrument response.

SYNTAX: TRANSFER {FROM type {options}} , {TO type {options}} , {FREQLIMITS f1 f2 f3 f4} , {PREWHITENING ON—OFF—n}

INPUT: FROM type : Remove the instrument type by deconvolution. The allowed instrument types and their options are listed in a table below.

TO type : Insert the instrument type by convolution. The allowed instrument types and their options are listed in a table below.

FREQLIMITS f1 f2 f3 f4 : This is a low- and high-pass taper that can be used to filter the spectrum. f1 and f2 specify the high-pass filter and corresponds to the frequencies over which the taper is applied. The taper is zero below f1 and unity above f2. f3 and f4 specify the low-pass filter and correspond to the frequencies over which the taper is applied. The taper is unity below f3 and zero above f4. The defaults provide no tapering for any realistic seismic signal. Note that the filter applied by TRANSFER is acausal. If it is important to preserve the character of signal onsets, you may want to use the default values for FREQLIMITS and filter the output signal using a single-pass filter of your design.

PREWHITENING {ON} : Turn on prewhitening in the time domain before spectral operations, and compensating dewhitening in the time domain after spectral operations. Initially, the option is off. If the user turns it on without specifying the order, it will default to 6, unless the order has been changed in the WHITEN command.

PREWHITENING OFF : Turn off prewhitening.

PREWHITENING n : Turn on prewhitening and change the prewhitening order to n.

Available instrument types: *ACC acceleration BBDISP Blacknest specification of Broadband Displacement BBVEL Blacknest specification of Broadband Velocity BENBOG Blacknest specification of Benioff by Bogert ****DBASE search database for applicable file:** EVALRESP, POLEZERO, or FAP **Note:** In order to use the DBASE type, the user must have access to an Oracle database with links to the applicable files, the database must be formatted as described below, and the user must have the Oracle version of sac2000. DSS LLNL Digital Seismic System DWWSSN Digital World Wide Standard Seismograph Station EKALP6 Blacknest specification of EKA LP6 EKASP2 Blacknest specification of EKA SP2 ELMAG Electromagnetic ****EVALRESP EVRESP** code by Thomas J. McSweeney ****FAPFILE** reads Frequency, Amplitude, Phase file GBALP Blacknest specification of GBA LP GBASP Blacknest specification of GBA SP GENERAL General seismometer GSREF USGS Refraction HFSLPWB Blacknest specification of HFS LPWB IW EYEOMG-spectral differentiation LLL LLL broadband analog seismometer LLSN LLSN L-4 seismometer LNN Livermore NTS Network instrument LRSMLP Blacknest specification of LRSM LP LRSMSP Blacknest specification of LRSM SP *NONE displacement, this is the default NORESS NORESS (NRSA) NORESSHF NORESS high frequency element OLDBB Old Blacknest specification of BB OLDKIR Old Blacknest specification of Kirnos ****POLEZERO** reads Pole Zero file PORTABLE Portable seismometer with PDR2 PTBLLP Blacknest specification of PTBL LP REDKIR Blacknest specification of RED Kirnos REFTEK Reftek 97-01 portable instrument RSTN Regional Seismic Test Network S750 S750 Seismometer SANDIA Sandia system 23 instrument SANDIA3 Sandia new system with SL-210 SRO Seismic Research Observatory *VEL velocity WA Wood-Anderson WABN Blacknest specification of Wood-Anderson WIECH Wiechert seismometer WWLPBN Blacknest specification of WWSSN long period WWSP WWSSN short period WWSPBN Blacknest specification of WWSSN short period YKALP Blacknest specification of YKA long period YKASP Blacknest specification of YKA short period

***Note:** ACC, VEL, and NONE do not refer to actual seismometer specifications but to acceleration, velocity, and displacement respectively. When these are specified as the TO type, IDEP is set accordingly.

****Note:** DBASE, EVALRESP, FAPFILE, and POLEZERO do not refer to actual seismometer specifications. They are described in greater detail below.

options : A set of zero or more of the following depending upon the **specific instrument type:** SUBTYPE subtype **the following types use the following subtypes:** **FAP:** name of file to be read **LLL:** LV, LR, LT, MV, MR, MT, EV, ER, ET, KV, KR, KT **LNN:** BB, HF **NORESS:** LP, IP, SP **POLEZERO:** name of file to be read **RSTN:** [CP, ON, NTR, NY, SD][KL, KM, KS, 7S][Z, N, E] **SANDIA:** [N, O][T, L, B, D, N, E][V, R, T] **SRO:** BB, SP, LPDE FREEPERIOD v **the following types use FREEPERIOD:** ELMAG, GENERAL, IW, LLL SUBTYPE BB, REFTEK (v must be 15.0 or 30.0 for ELMAG) MAGNIFICATION n **the following types use MAGNIFICATION:** ELMAG, GENERAL (n must be 375, 750, 1500, 3000, or 6000 for ELMAG) NZEROS n **the following types use NZEROS:** GENERAL, IW DAMPING v **the following types use DAMPING:** GENERAL, LLL SUBTYPE BB, REFTEK CORNER v **the following types use CORNER:** LLL SUBTYPE BB, REFTEK GAIN v HIGHPASS v **the following types use HIGHPASS:** REFTEK

The following options are used with the EVRESP option. FNAME filename STATION sta CHANNEL chan NETWORK ntwk DATE date TIME time

DEFAULT VALUES: TRANSFER FROM NONE TO NONE FREQUENCY -2. -1. 1.E5 1.E6

DESCRIPTION: This command can be used to alter the instrument response of seismic data. It is a modification of the program TRANSFER developed by Keith Nakanishi. A frequency domain deconvolution by spectral division is used to remove an instrument response, and a frequency domain convolution by spectral multiplication is used to insert a new instrument response.

Most of the implementation is done using double-precision (64-bit) arithmetic. If the FROM instrument type is NONE, then no instrument is removed, and the original trace is presumed to be a displacement. This is useful for adding instrument responses to synthetic seismograms. If TO type is NONE, then no instrument is inserted.

Many of the instruments have options which further specify the response. The most common of these options is the instrument subtype. A few instruments require that certain numerical parameters be specified, and do not use the subtype option. For a list of instruments, and a list of the instruments that use subtypes or other parameters, see the tables below.

Optional frequency domain cosine tapering can be used to select a bandlimited response. Optional prewhitening can also be used to flatten the spectrum of the input time series before transforming in the frequency domain. This should reduce the dynamic range of the spectral values, and improve the accuracy of the overall operation at high frequencies for seismic data.

The header field SCALE is set to 1.0, to prevent redundant scaling.

Unless other units are expected for a given instrument, the TRANSFER's returned values will be in nanometers (or nm/sec, or nm/sec/sec, etc). When the EVRESP option is used, the values are converted to nm before they are returned to the user. In versions 58 and 58a-58e, there was an inconsistency; values were being returned in nm except in the case of EVALRESP, which returned the values in meters. In addition, the NANO option was provided which, when turned on, would multiply the values by 1e09 before returning. This option had the effect of converting meters to nm, but the unintended consequence was that it converted nm to something much much smaller. In version 59, this inconsistency has been removed, and so has the NANO option.

EXAMPLES: To remove the instrument response from the RSTN station NYKM.Z and **apply the instrument response for DSS without pre-whitening:**

u: READ NYKM.Z

u: TRANS FROM RSTN SUBTYPE NYKM.Z TO DSS PREW OFF

To remove the LLL broadband instrument response and apply the SRO instrument **response with frequency tapering and prewhitening**:

u: READ ABC.Z

u: TRANS FROM LLL TO SRO FREQ .02 .05 1. 2. PREW 2

The passband of the resulting trace will be flat from .05 Hz to 1 Hz and will be zero below .02 Hz and above 2 Hz. Prewhitening of order 2 is applied in the time domain before deconvolution and the effect is removed in the time domain after convolution. To transfer from the electromagnetic instrument response to **displacement**:

u: READ XYZ.Z

u: TRANSFER FROM ELMAG FREEP 15. MAG 750. TO NONE

POLEZERO OPTION: One of the instrument types is called POLEZERO.

This option uses the Omega (Omega\) convention. This type lets you describe a general instrument response by specifying a file which contains its poles and zeros. The options in the file are keyword driven and the numbers are in free format. You may specify a multiplicative scaling constant by putting a line in the file containing the keyword “CONSTANT” followed by a floating point number. The default for this constant is 1.0 if you omit this line. You specify the number of poles by putting a line in the file with the keyword “POLES” following by an integer number. The next lines in the file until another keyword is read become the poles for this instrument. Each such line contains two floating point numbers specifying the real and imaginary parts of one of the poles. If you have fewer lines specifying poles than you stated on the “POLES” line, the remaining poles are assumed to lie at the origin. You specify the zeros in the same way with a “ZEROS” keyword line following by lines specifying the zeros that do not lie at the origin. You may specify up to 15 poles and 15 zeros. For example, the following is the specification for **the SRO broadband seismometer**:

ZEROS 4

-0.125 0.0

-50.0 0.0

POLES 4

-0.13 0.0

```
-6.02 0.0  
-8.66 0.0  
-35.2 0.0  
CONSTANT -394.0
```

Notice that since two of the zeros are at the origin, they don't have to be specified in the file. Also notice that the options may appear in any order in the file. To use this option you specify the type to be POLEZERO and the subtype to be the name of the file. This may be a file in the current directory or in some other directory if you specify the absolute or relative pathname. It may also be the name of a global file contained in the "polezero" subdirectory of the "sacaux" directory. By putting a file in this global directory, anyone on your system can easily use it. Nakanishi, K., "Computer code for the transfer function of seismic systems", Lawrence Livermore National Lab., UCID-18071, 1979.

EXAMPLE: suppose the file was named sro.pz and you want to remove the instrument response from station ABC.Z.

```
u: READ ABC.Z
```

```
u: TRANSFER FROM POLEZERO SUBTYPE SRO.PZ TO NONE
```

EVALRESP OPTION: This option enables the application of transfer functions extracted from SEED data volumes using the evresp code (Version 3.2.x) by Thomas J. McSweeney or rdseed (Version 4.1.x). The RESP files must be in the current directory or must be specified by full path and name.

To identify the correct RESP file and to extract the proper transfer function from that file, EVALRESP uses information from the SAC headers. The fields are station (KSTNM), channel (KCMPNM), date and time (KZDATE & KZTIME), network (KNETWK), and location ID.

Location ID is referred to as LOCID; it distinguishes between multiple seismometers with the same station and channel names, operating at the same time. Data received from IRIS in SAC format (or converted to SAC with RDSEED) will have KHOLE set to a valid LOCID if one is necessary. If the user is informed of real LOCIDs in the EVALRESP file, the user can set KHOLE with CH (HELP CH for details). SAC will use KHOLE as LOCID if it is a two character alpha-numeric string (padded with spaces or

not). At the present time (7/19/2000), LOCID is useful in a small number of cases.

It is possible to override the header values by specifying additional options **to EVALRESP. The possible options are:**

STATION CHANNEL NETWORK DATE TIME LOCID FNAME

and each option must be followed by an appropriate value. If DATE is not set in the header and is not specified as an option, then the current date is used in the search. If TIME is not set in the seismogram header and is not specified as an option, then the current system time is used in the search. If network is not specified, then the search for a transfer function defaults to use any network. If LOCID is not set at the command line or in KHOLE, then the search for the transfer function defaults to use any LOCID. If TYPE is not specified and is not set in the seismogram header then a velocity transfer function is computed. To force TRANSFER to use a specific SEED response file use the FNAME option followed by the filename.

EXAMPLES: To remove the instrument response from the seismogram in memory (assuming **a response file exists**):

u: TRANSFER FROM EVALRESP

To remove the instrument response from 16.42.05.5120.TS.PAS.BHZ.SAC **and apply the response from station COL, channel BHZ for the same time period:**

u: r 16.42.05.5120.TS.PAS.BHZ.SAC **u:** TRANSFER FROM EVALRESP TO EVALRESP STATION COL

To plot the instrument response in units of displacement for station COL, **channel BHZ, network IU, for the date 1992/02 and time 16:42:05:**

u: funcgen impulse npts 16384 delta .05 begin 0 **u:** transfer to evalresp station COL channel BHZ network IU type DIS date & **1992/2 time 16:42:05 u:** fft **u:** psp am (**NOTE:** the & only indicates a continued line in the documentation.)

To remove the instrument response from 16.42.05.5120.TS.PAS.BHZ.SAC **using a response contained in file /tmp/Responses/RESP.TS.PAS.BHZ:**

u: r 16.42.05.5120.TS.PAS.BHZ.SAC **u:** TRANSFER FROM EVALRESP FNAME /tmp/Responses/RESP.TS.PAS.BHZ TO NONE

DBASE OPTION: Note: In order to use the DBASE type, the user must have access to an Oracle database with links to the applicable files, the database must be formatted as described below, and the user must have the Oracle version of sac2000.

This option enables the deconvolution of transfer functions extracted from SEED response files using the evresp Version 3.2.6 code by Thomas J. McSweeney. The response files are assumed to have been extracted from SEED (VERSION 4.1.x), and the locations of the appropriate files must be stored in an accessible Oracle database using the INSTRUMENT and SENSOR tables of the CSS 3.0 schema. (See help for the READDB command for more information on the database **requirements.**) **Fields which must be set are:** instrument.inid instrument.dir instrument.dfile instrument.rsptype sensor.sta sensor.chan sensor.time sensor.endtime sensor.inid

The instrument.rsptype fields must contain the string 'evresp' and the remaining fields must be set as appropriate to join and point to the correct dir and dfile given the correct station, channel, and time. The DBASE option can only be used after the keyword FROM. In other words, TRANSFER FROM DBASE is OK but TRANSFER TO DBASE is not. If the database query is unsuccessful, TRANSFER will attempt to find an appropriate response file in the current directory. If that fails, the seismogram is not modified.

FAPFILE OPTION: This option is similar to the POLEZERO option, but instead reads a Frequency - Amplitude - Phase (FAP) file in the standard format used at the Center for Monitoring Research (pIDC). A (partial) example of such a file is shown below.

```
# # Displacement response for Array # # Example: ST01 z # #
Geotech 23900 seismometer # # Phase unwrapped # theoretical 0 in-
strument fap Organization 40 0.100000 1.576582e-01 -52.923801 0.000000
0.000000 0.125990 3.511520e-01 -61.669102 0.000000 0.000000 0.200000 1.634426e+00
-79.966599 0.000000 0.000000 0.368400 1.171214e+01 -107.522003 0.000000
0.000000 0.500000 3.135000e+01 -126.447998 0.000000 0.000000 0.683990
8.322500e+01 -155.035004 0.000000 0.000000 0.800000 1.273452e+02 -174.207001
0.000000 0.000000
```

In this format, lines starting with a “#” sign are comments. **The line:** theoretical 0 instrument fap Organization is ignored by SAC.

The next line (“40”) is the number of fap lines to follow. The remaining lines are frequency, amplitude, phase, amplitude error, and phase error.

The error columns are not used by SAC, but must be present in order for the file to be parsed correctly.

EXAMPLE: suppose the fap file was named ABC.PAZ and you want to remove the instrument response from station ABC.Z.

u: READ ABC.Z

u: TRANSFER FROM FAPFILE SUBTYPE ABC.PAZ TO NONE

SEE COMMANDS: WHITEN

ACKNOWLEDGEMENTS: Roger Hanscom did the original conversion of Keith Nakanishi’s TRANSFER program. George Randall added the prewhitening option and was a major contributor to the testing and documentation of this command.

LATEST REVISION: May 06, 1998 (Version 00.57)

3.15 CND: Conditional Execution Module

3.15.1 Break

3.15.2 Do

3.15.3 Else

3.15.4 Elseif

3.15.5 Enddo

3.15.6 Endif

3.15.7 If

3.15.8 While

3.16 NNM: Neural Network Module

3.16.1 Writenn

3.17 XYZ: XYZ (3-d) Data Processing Module

3.17.1 Contour

SUMMARY: Produces contour plots of data in memory.

SYNTAX: CONTOUR {ASPECT ON—OFF}

INPUT: ASPECT {ON} : Turn aspect ratio option on. When this option is on, the viewport of the contour plot will be adjusted to maintain the y to x aspect ratio of the data.

ASPECT OFF : Turn aspect ratio option off. When off, the full viewport is used.

DEFAULT VALUES: CONTOUR ASPECT OFF

DESCRIPTION: This command can be used to produce a contour plot of the of any other two-dimensional array data, including the output of the SPECTROGRAM command. The SAC data plotted by this command must of of file type “XYZ” (SAC header variable IFTYPE set to “IXYZ”). Several commands control how the data is **displayed:** ZLEVELS for the spacing and number of contour levels, ZLINES for linestyles, ZLABELS for contour labeling, ZTICKS for directional tick marks, and ZCOLORS for line colors. Depending upon the contouring options selected, two different contouring algorithms are used. A fast scan method is used if no only solid linestyles are selected and no tick marks or labels are requested. Otherwise, a slower method, where entire line segments are first assembled before they are drawn, is used. You may want to use the fast scan method for a quick look at your data and then select other options for a final version.

EXAMPLES: In the first example (shown below) a file is read and contoured using default values.

EXAMPLES (cont.): In this example, the same file is read and the header is listed to determine the range of the z data (DEPMIN and DEPMAX.) Only selected portions of the output from LISTHDR are shown. A range of contour levels between 700 km and 1150 km and an increment of 25 km is selected. A list of four linestyles is selected, starting with a solid line. The list will be repeated for every four contour levels. A title is defined and the **contour plot was generated:**

```
u: READ MYDATA
u: LISTHDR
s: FILE: MYDATA
s: NPTS = 10000
s: IFTYPE = GENERAL XYZ (3-D) FILE
s: DEPMIN = 697.71
s: DEPMAX = 1154.4
s: NXSIZE = 100
s: XMINIMUM = 82574.
s: XMAXIMUM = 86992.
s: NYSIZE = 100
s: YMINIMUM = 0.47439E+06
s: YMAXIMUM = 0.47720E+06
u: ZLEVELS RANGE 700 1150 INCREMENT 25
u: ZLINES LIST 1 2 3 4
u: TITLE 'Katmai topography from survey data [inc = 25 km]'
u: CONTOUR
```

The result of this example is shown in the figure below.

EXAMPLES (cont.): In the final example, the same data is used but different display options are selected. Integer labels are selected for every fourth contour level and “down” tick marks are selected for the contour levels in between. Solid linestyles are used for all contour levels.

```
u: READ MYDATA
u: ZLEVELS RANGE 700 1150 INCREMENT 25
```

```

u: ZLABELS ON LIST INT OFF OFF OFF
u: ZTICKS ON LIST 0 -1 -1 -1
u: ZLINES LIST 1
u: TITLE 'Katmai topography from survey data [labels and ticks]'
u: CONTOUR

```

The result of this example is shown in the figure below.

ACKNOWLEDGEMENTS: The fast scan contouring subroutine was developed by Dave Harris.

HEADER VARIABLES: REQUIRED: ,BEGINREVISIONBARS
 IFTYPE (set to "IXYZ"), NXSIZE, NYSIZE
,ENDREVISIONBARS : USED: XMINIMUM, XMAXIMUM, YMIN-
 IMUM, YMAXIMUM

SEE COMMANDS: ZCOLORS, ZLABELS, ZLEVELS, ZLINES, ZTICKS,
 SPECTROGRAM and the SAC User's Manual section on Writing SAC Data
 Files.

LATEST REVISION: JULY 22, 1991 (Version 10.6d)

3.17.2 Grayscale

SUMMARY: Produces grayscale images of data in memory.

SYNTAX: GRAYSCALE {options} **where options are one or more of the following:**

VIDEOTYPE NORMAL—REVERSED SCALE v ZOOM n XCROP n1
 n2—ON—OFF YCROP n1 n2—ON—OFF

SPECIAL NOTE: This command uses executables that are not distributed with SAC. To use this command you must first install the Utah Raster Toolkit. The Utah **Raster Toolkit can be obtained via anonymous FTP as follows:**

```

ftp cs.utah.edu
cd pub
get urt-3.0.tar.Z

```

If ARPAnet is not available, or if you have questions about the Utah Raster **Toolkit, send mail to:** toolkit-request\CS.UTAH.EDU (ARPA),

OR {ihnp4,decvax}!utah-cs!toolkit-request (UUCP).

INPUT: VIDEO NORMAL : Set video type to normal. In normal mode, data with near minimum values are black and data near maximum are white.

VIDEO REVERSED : Set video type to reversed. In reversed mode, data with near minimum values are white and data near maximum are black.

SCALE v : Change data scaling factor to v. The data is scaled by raising it to the vth power. Values less than one will smooth the image, reducing peaks and valleys. Values greater than one will spread the data.

ZOOM n : Image is increased to n times its normal size by pixel replication.

XCROP n1 n2 : Turn x cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.

XCROP {ON} : Turn x cropping option on and use previously specified cropping limits.

XCROP OFF : Turn x cropping option off. All of the data in the x direction is displayed.

YCROP n1 n2 : Turn y cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.

YCROP {ON} : Turn y cropping option on and use previous specified cropping limits.

YCROP OFF : Turn y cropping option off. All of the data in the y direction is displayed.

DEFAULT VALUES: GRAYSCALE VIDEOTYPE NORMAL SCALE 1.0 ZOOM 1 XCROP OFF YCROP OFF

DESCRIPTION: This command can be used to produce a grayscale image of the output of the SPECTROGRAM command or of any other two-dimensional array data. The SAC data displayed by this command must be of file type "xyz".

ANOTHER SPECIAL NOTE: SAC starts a shell script which runs the image manipulation and display programs and then displays the SAC prompt again. There is a delay, significant for large images and/or slower machines, before the image is actually displayed.

LIMITATIONS Images of 512 by 1000 are the maximum displayed.

ACKNOWLEDGEMENTS: This command was developed by Terri Quinn. The grayscale images are manipulated and display using the University of Utah's Raster Toolkit. The Utah Raster Toolkit and accompanying documentation; John W. Peterson, Rod G. Bogart, and Spencer W. Thomas.

HEADER VARIABLES: REQUIRED: : IFTYPE, NXSIZE, NY-SIZE

ERROR MESSAGES: SAC; getsun: Command not found. Several utility programs distributed with the Utah Raster Toolkit are required.

SEE COMMANDS: SPECTROGRAM

LATEST REVISION: March 22, 1990 (Version 10.5a)

3.17.3 Image

SUMMARY: Produces color sampled image plots of data in memory.

SYNTAX: IMAGE {COLOR—GREY} {BINARY—FULL} {PRINT {pname} }

INPUT: COLOR—GREY : Produce a color or greyscale image.

BINARY—FULL : Produce an image where all positive values plot in one color and all negative values plot in a second color, or plot the full range of the data.

PRINT {pname} : Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES: IMAGE COLOR FULL

DESCRIPTION: The image command allows the user to make color or grayscale images from a SAC 3-D data file such as those generated by the spectrogram, scallop, or bbfk commands. It can also be used to plot imported data provided they are in the SAC 3-D data format. Different sections of the image can be viewed using the xlim and ylim commands and amplitudes can be scaled using the usual unary operations provided in SAC.

HEADER VARIABLES: REQUIRED: : IFTYPE (set to "IXYZ"),

NXSIZE, NYSIZE

USED: : XMINIMUM, XMAXIMUM, YMINIMUM, YMAXIMUM

LATEST REVISION: May 26, 1995 (Version 00.31)

3.17.4 Sonogram

SUMMARY: Calculate a spectrogram equal to the difference between two smoothed versions of the same spectrogram.

SYNTAX: SONOGRAM options **where options are one or more of the following:**

WINDOW *v* SLICE *v* ORDER *n* CBAR {ON—OFF} YMIN *v* YMAX
v FMIN *v* FMAX *v* BINARY—FULL METHOD {PDS—MEM—MLM}
{COLOR—GRAY} PRINT {pname}

INPUT: WINDOW *v* : Set the sliding data window length in seconds to *v*. This window length determines the size of the fft.

SLICE *v* : Set the data slice interval in seconds to *v*. A single spectrogram line is produced for each slice interval.

ORDER *n* : Specifies the number of points in the autocorrelation function used to compute the spectral estimate.

CBAR {ON—OFF} : Turn reference color bar on or off.

BINARY—FULL : Produce a binary image, or a full color image.

YMIN *v* : Specifies the minimum frequency to plot.

YMAX *v* : Specifies the maximum frequency to plot.

FMIN *v* : Specifies the smallest bandwidth over which each slice in the spectrogram will be smoothed.

FMAX *v* : Specifies the maximum bandwidth over which each slice in the spectrogram will be smoothed.

METHOD {PDS—MEM—MLM} : Specifies the type of spectral estimator used. MLM stands for maximum likelihood and MEM stands for maximum entropy spectral estimators, respectively. See description and references below.

{COLOR—GRAY} : Specifies a color or grayscale image.

PRINT {pname} : Prints the resulting plot to the printer named in

pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES: SONOGRAM WINDOW 2 SLICE 1 METHOD MEM ORDER 100 YMIN 0 YMAX FNYQUIST FMIN 2.0 fmax 6.0 full color

DESCRIPTION: The sonogram command computes a spectrogram equal to the difference between two smoothed version of the same spectrogram. Depending on the choice of smoothing parameters, fmin and fmax, the resulting spectrogram can enhance small amplitude spectral features that are more difficult to observe in a conventional spectrogram. This is particularly useful when looking for features like high frequency spectral modulations in seismic signals from mine blasts (c.f., Hedlin, 1990, Wuster, 1993).

LIMITATIONS: The size of the image in the frequency direction is 512.

PROBLEMS: There is currently very little error checking of the headers to make sure that they are from the same component and are contiguous in time. This will be corrected in the future.

HEADER VARIABLES: REQUIRED: : DELTA

CHANGED: : NPTS, DELTA, B, E, IFTYPE, DEPMIN, DEPMAX, DEPMEN

CREATED: : NXSIZE, XMINIMUM, XMAXIMUM, ,BREAK NY-SIZE, YMINIMUM, YMAXIMUM

LATEST REVISION: May 26, 1995 (Version 00.31)

3.17.5 Spectrogram

SUMMARY: Calculate a spectrogram using all of the data in memory.

SYNTAX: SPECTROGRAM options **where options are one or more of the following:**

WINDOW v SLICE v ORDER n CBAR {ON—OFF} {SQRT—NLOG—LOG10—NOSCALING}
YMIN v YMAX v METHOD {PDS—MEM—MLM} {COLOR—GRAY}
PRINT {pname}

INPUT: WINDOW v : Set the sliding data window length in seconds to v. This window length determines the size of the fft.

SLICE v : Set the data slice interval in seconds to v. A single spectrogram line is produced for each slice interval.

ORDER n : Specifies the number of points in the autocorrelation function used to compute the spectral estimate.

CBAR {ON—OFF} : Turn reference color bar on or off.

{SQRT—NLOG—LOG10—NOSCALING} : Specify natural log, log base 10, or square root scaling of amplitudes.

YMIN v : Specifies the minimum frequency to plot.

YMAX v : Specifies the maximum frequency to plot.

METHOD {PDS—MEM—MLM} : Specifies the type of spectral estimator used. MLM stands for maximum likelihood and MEM stands for maximum entropy spectral estimators, respectively. See description and references below.

{COLOR—GRAY} : Specifies a color or grayscale image.

PRINT {pname} : Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES: SPECTROGRAM WINDOW 2 SLICE 1 METHOD MEM ORDER 100 NOSCALING YMIN 0 YMAX FNYQUIST COLOR

DESCRIPTION: A spectrogram is computed by calculating power spectra of consecutive, possibly overlapping time windows of data and plotting the spectra side by side along a time axis. The spectra are calculated from a truncated autocorrelation function using either the maximum likelihood method (MLM), maximum entropy method (MEM), or Power Density Spectral method (PDS). In general, the high resolution, maximum likelihood and maximum entropy methods are preferred because they improve resolution and because they do not produce artifacts (sidelobes) in the spectra due leakage of energy between different frequencies. Descriptions of these techniques can be found in Kanasewich (1981) and Lacoss (1971) and the references therein. The length of the truncated autocorrelation function is determined by the order parameter. To maintain consistency with the

spe subroutines we have set the defaults order to 200 for the power density spectra (pds) and 100 for the maximum entropy and maximum likelihood spectral estimates. In sac the length of each data window is determined by the window parameter. The spacing between spectra along the spectrograms time axis is determined by the slice parameter. The difference between these two parameters determines the amount of overlap between adjacent time window as indicated in the diagram below.

Time — 0 1 2 3 4 5 6 7 8 9 10 11 —....—....—....—....—....—....—....—....—....—
 —----— window 1, First time in spectrogram will be at the center of this
 window. —----— window 2 —----— window 3
 —..— The slice is the difference between the start times of adjacent
 windows. ... —----— —----— —----—

The start and end points on the spectrograms time axis depend on the length of the time series being analysed and the window and slice parameters. The spectrogram's start time is one-half a window later than the time series start time because it is defined as the center of time of the first window. SAC doesn't pad the start of the data with zeros.

Kanasewich, E. R., "Time Sequence Analysis in Geophysics", The University of Alberta Press, Edmonton, 1981.

Lacoss, R. T., "Data Adaptive Spectral Analysis Methods", Geophysics, Vol. 36, 661-675, 1971.

LIMITATIONS: The size of the image in the frequency direction is 512.

PROBLEMS: There is currently very little error checking of the headers to make sure that they are from the same component and are contiguous in time. This will be corrected in the future.

HEADER VARIABLES: REQUIRED: : DELTA

CHANGED: : NPTS, DELTA, B, E, IFTYPE, DEPMIN, DEPMAX, DEPMEN

CREATED: : NXSIZE, XMINIMUM, XMAXIMUM, ,BREAK NY-SIZE, YMINIMUM, YMAXIMUM

LATEST REVISION: May 26, 1995 (Version 00.31)

3.17.6 Zcolors

SUMMARY: Controls the color display of contour lines.

SYNTAX: ZCOLORS {ON—OFF} {options} **where options is currently limited to:**

LIST c1 c2 ... cn

More options will be added in the future.

INPUT: ON : Turn color display of contour lines on.

OFF : Turn color display of contour lines off.

LIST c1 c2 ... cn : Set the list of contour color names to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the length of this list, the entire list is repeated.

cn : The name of a color from SAC's current color table.

DEFAULT VALUES: ZCOLORS OFF LIST RED GREEN BLUE

SEE COMMANDS: CONTOUR, COLOR

LATEST REVISION: April 30, 1990 (Version 10.5b)

3.17.7 Zlabels

SUMMARY: Controls the labeling of contour lines with contour level values.

SYNTAX: ZLABELS {ON—OFF} {options} **where options are one or more of the following:**

SPACING v1 {v2 {v3} } SIZE v ANGLE v LIST c1 c2 ... cn

SPECIAL NOTE: The LIST option must appear last in this command.

INPUT: ON : Turn labeling of contour lines on.

OFF : Turn labeling of contour lines off.

SPACING v1 {v2 {v3} } : Set the minimum, optimum, and maximum spacing between adjacent labels (in viewport coordinates) to v1, v2, and v3 respectively. If the second or third values are omitted, the previous values are used.

SIZE v : Set the size (height) of the labels (in viewport coordinates) to v.

ANGLE v : Set the desired maximum text angle the labels (in degrees from horizontal) to v.

LIST c1 c2 ... cn : Set the list of contour labels to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the length of this list, the entire list is repeated.

cn : ON—OFF—INT—FLOATn—EXPn—text

ON : Place a label on corresponding contour line. Use Fortran's free format capabilities to format the label from the contour level value.

OFF : Do not place a label on corresponding contour line.

INT : Place an integer label on corresponding contour line.

FLOATn : Place a floating point label on corresponding contour line with n values to the right of the decimal point. If n is omitted, the previous value for n is used.

EXPn : Place an exponentially formatted label on corresponding contour line with n values to the right of the decimal point. If n is omitted, the previous value for n is used.

text : Use text to label the corresponding contour line.

DEFAULT VALUES: ZLABELS OFF SPACING 0.1 0.2 0.3 SIZE 0.0075 ANGLE 45.0 LIST ON

EXAMPLES: See CONTOUR for examples of the use of ZLABELS.

SEE COMMANDS: CONTOUR

LATEST REVISION: April 30, 1990 (Version 10.5b)

3.17.8 Zlevels

SUMMARY: Controls the contour line spacing in subsequent contour plots.

SYNTAX: ZLEVELS {options} **where options are one or more of the following:**

SCALE RANGE v1 v2 INCREMENT v NUMBER n LIST v1 v2 ... vn

INPUT: SCALE : Scale the range of the contour levels to the data.

RANGE v1 v2 : Set the range (minimum and maximum) of the contour levels to v1 and v2. You should use either the SCALE or the

RANGE option but not both.

INCREMENT v : Set the increment between contour levels to v.

NUMBER n : Set the number of contour levels to n. You should use either the INCREMENT or the NUMBER option but not both.

LIST v1 v2 ... vn : Set the list of contour levels to v1, v2, etc. All other options are ignored if you use this one.

DEFAULT VALUES: ZLEVELS SCALE NUMBER 20

EXAMPLES: See CONTOUR for examples of the use of ZLEVELS.

LIMITATIONS: The maximum number of contour levels is 40.

SEE COMMANDS: CONTOUR

LATEST REVISION: April 30, 1990 (Version 10.5b)

3.17.9 Zlines

SUMMARY: Controls the contour linestyles in subsequent contour plots.

SYNTAX: ZLINES {ON—OFF} {options} **where options are one or more of the following:**

LIST n1 n2 ... nn REGIONS v1 v2 ... vn

INPUT: ON : Turn display of contour lines on.

OFF : Turn display of contour lines off.

LIST n1 n2 ... nn : Set list of linestyles to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the number of entries in the list, the entire list is repeated.

REGIONS v1 v2 ... vn : Set list of contour regions. The length of this list should be one less than the linestyle list. Contour levels less than a contour region value are assigned the linestyle of the corresponding entry in the linestyle list. Contour levels above the last contour region value are assigned the value of the last entry in the linestyle list.

DEFAULT VALUES: ZLINES ON LIST 1

EXAMPLES: To set up contours which cycle between four different linestyles:

u: ZLINES LIST 1 2 3 4

To set contours with dotted lines representing levels below 0.0 and solid lines representing contours above 0.0:

u: ZLINES LIST 2 1 REGIONS 0.0

See CONTOUR for more examples of the use of ZLINES.

SEE: CONTOUR

LATEST REVISION: April 30, 1990 (Version 10.5b)

3.17.10 Zticks

SUMMARY: Controls the labeling of contour lines with directional tick marks.

SYNTAX: ZTICKS {ON—OFF} {options} **where options are one or more of the following:**

SPACING *v* LENGTH *v* DIRECTION DOWN—UP LIST *c1 c2 ... cn*

INPUT: ON : Turn tick mark labeling of contour lines on.

OFF : Turn tick mark labeling of contour lines off.

SPACING *v* : Set the spacing between adjacent tick marks (in viewport coordinates) on each line segment to *v*.

LENGTH *v* : Set the length of each tick mark (in viewport coordinates) to *v*.

DIRECTION DOWN : Tick marks point in the direction of decreasing *z* value.

DIRECTION UP : Tick marks point in the direction of increasing *z* value.

LIST *c1 c2 ... cn* : Set the list of contour ticks marks to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the length of this list, the entire list is repeated. A value of ON means that tick marks are placed on that contour line. A value of OFF means that no tick marks are placed on that contour line. line.

DEFAULT VALUES: ZTICKS OFF SPACING 0.1 LENGTH 0.005
DIRECTION DOWN LIST ON

EXAMPLES: See CONTOUR for examples of the use of ZTICKS.

SEE COMMANDS: CONTOUR

LATEST REVISION: April 30, 1990 (Version 10.5b)

3.18 FKS: frequency-wavenumber (k) spectral analysis

3.18.1 Arraymap

SUMMARY: Produces a map of the array or “coarray” using all files in SAC memory.

SYNTAX: ARRAYMAP ARRAY — COARRAY

INPUT: ARRAY : This option maps the offsets X and Y, assumed to have been set up in the SAC header (see the HEADER DATA section below).

COARRAY : This option plots delta X and delta Y for all pairs of stations.

DEFAULT VALUES: ARRAYMAP ARRAY

FUNCTIONAL MODULE: FK Spectrum (fks)

HEADER DATA: The following header variables must be set up in advance, using the SAC macro WRXYZ, or its functional equivalent. All offsets are measured in kilometers from a reference location. **USER7 :** Contains easterly offset (x).

USER8 : Contains northerly offset (y). The upward offset (z) is not used by this command.

LIMITATIONS: Maximum number of stations allowed in BBFK command.

SEE COMMANDS: WRXYZ. This is a SAC macro; It can be found in the global SAC macro directory, SAC AUX/macros . Documentation provided in the macro. BBFK July 22, 1991 (Version 10.5c)

3.18.2 Bb fk

SUMMARY: Computes the broadband frequency-wavenumber (FK) spectral estimate, using all files in SAC memory.

SYNTAX: BBFK {FILTER} {NORMALIZE} {EPS v} {MLM — PDS} {EXP n} {WAVENUMBER v} {SIZE m n} {LEVELS n} {DB} {TITLE text} {WRITE {ON — OFF} fname} {SSQ n} {PRINT {pname} }

INPUT: FILTER : Apply the bandpass filter designed in the most recent FILTERDESIGN command.

NORMALIZE : Normalizes the covariance matrix with the Capo method. A good idea if the signals vary much in amplitude among channels.

EPS v : Regularization quantity for covariance matrix. Diagonal matrix entries are multiplied by $(1.0 + \text{EPS})$.

MLM : Use maximum likelihood method for high-resolution estimate.

PDS : Take power density spectra without maximum likelihood method.

EXP n : Power to which the wavenumber spectrum will be raised.

WAVENUMBER v : Number of waves from which to sample spectral estimates.

SIZE m n : Size of contour plot in polar mode: m is an even num of plot samples in the azimuth direction; n is an even num of plot samples in the wavenumber direction (m*n is limited to 40,000).

LEVELS n : Number of contour intervals.

DB : Log scaling of plot in decibels.

TITLE text : Title used in plot.

WRITE {ON — OFF} fname : Whether to compute & write contour data in square mode to disk (as a type xyz SAC file). fname is file or path name (may be an absolute or relative). If no filename has been specified, the default is “BBFK”. ON will reactivate fname most recently used. OFF turns writing off.

SSQ n : Size of the square (number of samples taken along each margin of the square). Maximum allowed is 200.

PRINT {pname} : Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES: BBFK EPS .01 PDS EXP 1 WVENUMBER 1.0 SIZE 90 32 LEVELS 11 WRITE OFF SSQ 100 (SSQ matters only if WRITE has been positively specified).

DESCRIPTION: The BBFK command allows the user to compute broadband frequency wavenumber spectra. It is based on the work of

NAWAB et al., 1985 and many other references in the seismic and engineering literature.

Nawab, SH, FU Dowla, and RT Lacoss, Direction determination of wide-band signals, **IEEE Trans. Acous. Speech Sig. Proc.**, **33**: (5), 1114-1122, 1985

FUNCTIONAL MODULE: FK Spectrum (fks)

HEADER DATA: The following logic is used to determine how to choose or **calculate station/event offsets**:

Case 1: If a reference station is set in KUSER1 and is the same for all files, and USER7 and USER8 are set for all files, USER7 and USER8 are used as offsets.

Case 2: If station latitude (STLA) and station longitude (STLO) are set for all files, offsets are calculated using these, using the first file as the reference station.

Case 3: If USER7 and USER8 are set for all files, they are used as offsets.

Case 4: If event latitude (EVLA) and event longitude (EVLO) are set for all files then these are used to calculate offsets, using the first station as the reference station.

OUTPUT: The polar output is plotted immediately (not retained), the square output if requested is written out to disk. The FK peak, back azimuth and wavenumber are written to blackboard variables BBFK_AMP, BBFK_BAZIM and BBFK_WVNBR respectively.

ERROR MESSAGES: Size m or n not an even number. Offsets X,Y,Z not set in USER7,8,9 of headers. Coefficients produced by FILTERDESIGN not found, or filter type used was not "BP".

LIMITATIONS: The maximum number of stations allowed is 100. The maximum size of polar contour plot is $m \times n = 40,000$. The maximum size of square contour output is $i = 200$.

SEE COMMANDS: MAP: for plotting stations in an array, according to X,Y offsets stored in SAC header variables USER7 & USER8. July 22, 1991 (Version 10.6c)

3.18.3 Beamform

3.18.4 Gmtmap

3.19 MAT: matlab analysis routines

3.19.1 Closemat

3.19.2 Mat

SUMMARY: Copy SAC workspace into Matlab and either execute a user-specified m-file or else get a Matlab prompt for interactive manipulation. The SAC workspace is updated with changes made to the data after the return from Matlab.

SYNTAX: MAT [mfile]

DESCRIPTION: The mat command allows processing of SAC data from within SAC using the Matlab (Version 5) engine and any user-written m-files. When this command is executed, the SAC workspace is copied into **the following Matlab variables:** SeisData — an M-points by N-traces array of waveforms. SACdata — an M-element structure array containing the header information from the SAC workspace. BlackBoard — a structure array containing any blackboard variables.

SEISDATA: If the SAC data are time-domain, the SeisData array is real. Other wise it is complex. However, be aware that the default behavior of SAC's fft command is to produce transformed data in amplitude-phase format while in Matlab, the data will be treated as real-imaginary. The easiest way around that is to use the rlim option with SAC's fft.

You must return trace data from Matlab to SAC in the same domain as it was in before the mat command was executed. Otherwise, changes to the trace data made in Matlab will not be preserved. Also, you must not change the length of the traces in Matlab.

SACDATA: The SACdata structure array contains the following elements: times station event user descrip evsta llnl response trcLen

starting the Matlab engine. However, a Matlab license will be tied up while you are running SAC and this may inconvenience other users who cannot start a session. To exit the Matlab interpreter and/or close the engine, type “closemat” at either the SACMAT_{;;} or the SAC_; prompt.

HEADER CHANGES: Potentially all. User is responsible for consistency of changes.

EXAMPLE: Execute an m-file that converts the data to their absolute values. Assume the m-file is named absv.m and contains the one line SeisData=abs(SeisData); **u:** mat absv

NOTES: You may find it easier to develop a complex m-file directly from Matlab rather than from within the SAC-Matlab environment. The primary reasons are that there is no command line recall at the SACMAT_{;;} prompt and because SACMAT does not trap \hat{C} (used to stop errant m-files in Matlab). The easiest way to do this is to load your data into SAC, start the Matlab engine with mat, and then type save. This will save the workspace in a file called matlab.mat. You may then start a normal matlab session, and type load. This will load matlab.mat and you may then develop your application within Matlab.

The entire range of plotting commands are available. However, if you execute your m-file from SAC (i.e. mat mfilename) Matlab will return to SAC immediately after executing the last command in the m-file. Therefore, if you want to look at your plots created in Matlab either execute the m-file from the Matlab command line, or execute a pause in your Matlab script. . . . plot(SeisData) pause(10)

LATEST REVISION: Aug 9, 1997 (Version 00.56a)

3.19.3 Mat3c

3.19.4 Matdepmec

3.19.5 Recordsection

3.19.6 Setmat

3.20 CODA: Kevin Mayeda's coda magnitude

3.20.1 Mcoda

4 SAC Graphics File (SGF)

GFTOPS SUMMARY Converts an SGF file to a POSTSCRIPT formatted file. SYNTAX SGFTOPS sgf-file ps-file {width} {YES—NO} {scale} INPUT sgf-file : The complete name of a SAC Graphics File. ps-file : The desired name for the POSTSCRIPT file. width : A number specifying the width or thickness of lines in the output file. This is an integer in the range 1 to 8. Default is 1. A value of 3 would produce lines three times thicker than the default. See the figure below. YES—NO : Flag to apply additional scaling. A value of YES turns scaling on, NO ignores the additional scale parameter. scale : An additional scale factor. Can be used for making large or full-size maps. EXAMPLES To produce a POSTSCRIPT file call f001.ps from a SGF f001.sgf.

```
u: sgftops f001.sgf f001.ps
```

This file could then be sent to any printer capable of interpreting POSTSCRIPT commands. To produce this same POSTSCRIPT file where all lines are twice as thick as normal:

```
u: sgftops f001.sgf f001.ps 2
```

File Format

Overview Each SGF contains all the information needed to describe a single picture (called a frame.) The filenames are normally of the form “fnnn.sgf” where nnn is the three digit frame number. A translation program must be written to convert these files to the format needed for any specific graphics device.

Physical Format

A SGF contains variable length records with a maximum record size of 2500 32-bit words. The first 32-bit word of each record contains the length

of that record, including this word count. They are written in binary format for faster i/o. To keep them small and portable between different computer systems, all commands and data are stored in 16-bit integer format.

Draw Command The draw command (draw a line from the previous location to the new location) is the most common command. This command is simply a pair of integers giving the new x and y locations. These integers are in the range 0 to 32000 in the x direction and 0 to 24000 in the y direction. (This produces an aspect ratio of 3:4 which maps well to most output devices.)

Other Commands The rest of the commands (with one exception) consist of a command identification number, a data count, and zero or more data words. The identification number is a negative integer and tells the translation program what operation is to be performed. The use of negative integers makes it easy to distinguish these commands from the draw commands. The data count is the number of 16-bit data words contained in this command. This format allows for the future addition of new commands. Also it allows each translation program to quickly skip over commands that it cannot process. The one exception to this format is the null or no-op command. This has an identification number of -1 and contains no data count and no data words. It is used to fill out a record to an even number of 32-bit words. The table on the next page summarizes the current commands. A plot produced from a simple SGF is also included, along with a table describing the contents of that simple SGF.

Writing Your Own Conversion Program

It is not difficult to write a conversion program from the SGF format to your own graphics format. The best way is to start with one of the other SGF conversion programs described in this manual and modify it for your graphics library or device. See your system administrator or the person who installed SAC on your system for a copies of these conversion programs. If you want a program that interfaces with a graphics library, SGFPLOT is probably the best choice. For a laser printer or pen plotter conversion program, start with SGFTOPS. For a graphics terminal conversion program, try SGFTOTEK. Each of these programs are similar in structure and use subroutines from

the SAC library to handle the mechanics of user interaction, file i/o, and interpreting the SGF commands. By studying this section and using the appropriate SGF conversion program as a prototype, you can produce one for your system fairly easily.

SGF Commands Table ID Count Description -2 0 End of picture. -3 2 Move to the location contained in the two (x,y) data words. -4 1 Change color to value contained in data word. -5 * Write hardware text at current location. * Data count contains number of 16-bit words of text plus one. First data word is the number of characters in the text. Rest of data words contain the text, two characters per word. Last byte of last word is not significant if character count is odd. -6 2 Change hardware text size. Data words contain the text width and height as integer fractions of the maximum coordinate system size (32000). For example a value of 320 would set text size to 0.01 or one percent of the full plot size. -7 1 Change linestyle to value contained in data word. -8 1 Change the physical size of the plot. Data word is the desired length in the x direction in 0.001 inch increments. Default value is 10000 which is equivalent to 10.0 inches. None of the SGF conversion programs currently make use of this option.

FIGURE: [Click here to see an sample SGF plot.](#)

Contents of Sample SGF Table

Word Value Comment 1-2 20 32-bit word count of first buffer 3 -4 change color command 4 1 data count for color command 5 0 data value for color command 6 -7 change linestyle command 7 1 data count for linestyle command 8 1 linestyle of 1 = solid 9 -6 change hardware text size command 10 2 11 426 13 -3 move command 14 2 15 8800 x location of move = $8800/32000 = 0.275$ 16 4800 y location of move = $4800/24000 = 0.200$ 17 8800 x location of first draw 18 19200 y location of first draw 19 23200 x location of next draw 20 19200 y location of next draw 21 23200 22 4800 23 8800 24 4800 25 -7 change linestyle command 26 1 27 2 linestyle 2 28 -3 29 2 30 8800 31 19200 32 23200 draw 33 4800 34 -3 move 35 2 36 23200 37 19200 draw 38 8800 39 4800 40 -2 end of picture command 41 0 42 -1 no-op to fill to 32-bit word boundary

If you have technical questions about this page, contact: peterg@llnl.gov

– Peter Goldstein

This page maintained by: peterg@llnl.gov – Peter Goldstein

LLNL Disclaimer 8/12/98 UCRL-MA-112836

5 History

New Features and Releases

— v100.0 —

2005 February 23: Updated sgf plotting utilities to with enhanced versions from Arthur Snoke. New version of sgftops does translation rotation and scaling of postscript plots. sgftoops.csh is a csh shell script which creates encapsulated postscript that are compatible with latex and version 10 of Adobe Illustrator. The script requires epstool (<http://www.ghostgum.com.au/>) and ghostscript (<http://www.cs.wisc.edu/~ghost/>).

2004 December 8: Version 100.00 released for Solaris, Linux, and OS X

Recent enhancements and bug fixes:

Automatic byteswapping has been implemented. SAC will automatically read sac data files in either big or little endian format. (ie., It doesn't matter if your data were created on a little endian platform like intel based PC's or big endian platforms like Sun workstations.

readalpha/readtable (command for reading ascii tables or columns of xy data): a bug in the linux version has been fixed.

— v0.58 —

1) New Capabilities / Added Commands READCSS and WRITECSS have been rewritten from the ground up to provide complete CSS 3.0 compatibility. For details use HELP READCSS and HELP WRITECSS.

COMMIT, ROLLBACK, and RECALLTRACE (RECALL for short) are three new commands which add flexibility to the process of analyzing data. SAC now has two data storage locations in RAM, the original working memory that SAC has always had, and an I/O buffer. After reading a set of files (which puts identical information in both working memory and the I/O buffer), the user can process the files in working memory while leaving the I/O buffer untouched. The two locations contain information from the same

files, but the one is processed and the other is not. The user can choose from the following options: - ROLLBACK copies the unprocessed files from the I/O buffer to working memory, reverting to the previous state, - COMMIT copies the processed files from working memory into the I/O buffer, committing the changes in both locations, - RECALLTRACE rolls back the waveform certain header fields which are tightly linked to the waveform, while committing the remaining header fields; the user can filter a file, use PPK to get picks, and use RECALLTRACE to get the original waveform back but keep the picks. For more information use HELP COMMIT, HELP ROLLBACK, and HELP RECALLTRACE.

TRAVELTIME TAUP: TRAVELTIME now takes the TAUP option, specifying that the named file is of the format output by taup_curve: a traveltime estimator by Philip Crotwell at the University of South Carolina. The PICKS option saves requested picks in specified header fields T0-T9. Use HELP TRAVELTIME for details.

TRANSFER now can read response information from a Frequency/Amplitude/Phase (FAP) file. Also, the DBASE option sends SAC searching the Oracle database for pertinent EVALRESP, FAP, or POLEZERO files for instrument response. Use HELP TRANSFER.

SORT allows the user to sort the files in memory according to the header fields. It can sort in ascending or descending order, and can sort on as many as five header fields at once. For details, use HELP SORT.

CUTIM allows the user to cut the file in memory. Also allows the user to cut multiple segments of a file into multiple files. For details use HELP CUTIM.

READGSE and READSUDS were added to read the GSE2.0 data format, and the SUDS data format. For details use HELP READGSE and HELP READSUDS.

WRITEGSE now writes data in the GSE2.0 format. For details use HELP WRITEGSE.

READ now take a SEG Y option allowing it to read segy files of the type supported by IRIS/PASSCAL. For details use HELP READ.

READ now takes an ALPHA option to allow it to read SAC formatted

alphanumeric files such as those written with WRITE ALPHA. To avoid some of the confusion, the READALPHA command (which doesn't read SAC formatted alphanumeric files, but tabular column data) has been renamed READTABLE (or RTAB for short). For details, use HELP READ or HELP READTABLE.

WHITE (or PREWHITE) has been added as a command callable from either SAC's main shell, or from the SPE subprocess. It performs the same processing as the PREWHITEN option of the COR command. Use HELP WHITEN for details.

FILTERDESIGN is working again. FILTERDESIGN plots a graphical representation of a filter, the parameters of which are entered by the user at the command line. The FILE option saves Impulse Response, Group Delay, and Amplitude and Phase information in SAC files. Use HELP FILTERDESIGN for details.

PRINT option has been added to the following commands: BEGINFRAME, PLOT, PLOT1, PLOT2, BBFK, FILTERDESIGN, PLOTALPHA, PLOTCOR, PLOTDY, PLOTTPM, PLOTRECORDSECTION, IMAGE, SONOGRAM, and SPECTROGRAM. When used with the BEGINFRAME command, it signals the ENDFRAME command to print the resultant plot. All other commands simply print the resultant plot. It optionally takes the name of a printer as a parameter. It will write SGF files in the /tmp directory. Use the HELP command on any of the listed commands for details.

PRINT command will print the most recent SGF file produced. This presumes an SGF file has been produced since SAC2000 was booted. For details use HELP PRINT.

The SGF command now takes an OVERWRITE option. When OVERWRITE is on, each SGF file written overwrites the previous SGF file. This saves space on the hard disk, but the files disappear.

PRINTHELP hardcopies of help files. Use HELP PRINTHELP for details.

2) Important Changes The maximum number of files that can be read into SAC2000 has been increased from 200 to 1000.

Now that there are two memory locations where each file is stored, it is important for certain SAC commands, that the two copies of each file be identical. For this reason, the following commands will automatically perform either a COMMIT, ROLLBACK, or RECALLTRACE before executing: MERGE DATAGEN MORE READ MORE READTABLE MORE (formerly READALPHA) READCSS MORE READDB MORE READGSE MORE READHDR MORE READSDD MORE READSUDS MORE SORT WRITE WRITECSS WRITEHDR WRITESP WRITESTACK In each case, COMMIT is the default, but options allow the user to change to ROLLBACK or RECALLTRACE. Use the HELP command for further information.

The commands MULF, DIVF, ADDF, and SUBF have – in previous versions – left the user with the headers from the file being added (or what have you) in, as opposed to keeping the original header. This version and future versions will keep the original header by default. An option has been provided to allow the user to keep the new header instead. For details use HELP MULF, HELP DIVF, HELP ADDF, or HELP SUBF.

DELETECHANNEL now accepts a range of filenumbers using the dash (-). For details, use HELP DELETECHANNEL.

The MERGE command has been updated to allow files with overlapping data points to merge as long as the overlapping points line up exactly.

PLOT2 now plots spectral files (IRLIM and IAMPH) as relative whether the RELATIVE or ABSOLUTE option is selected. This is because frequency is not measured on an arbitrary scale as is time. For details use HELP PLOT2.

sacio.a is a new library available on our ftp site which is a whole lot smaller than sac.a and allows full use of all the SAC I/O routines written up in the SAC manual. These routines can be called transparently from either C or FORTRAN code. Use HELP INPUT_OUTPUT and HELP APPENDIX for details.

3) Bug Fixes Modified SPE subprocess to allow windows greater than 2048.

Fixed a bug in e1 compressed data expansion

Fixed a bug that prevented SAC from reading from links.

— v0.57 —

0) New Area Code On March 14, 1998, the area code for Lawrence Livermore National Laboratory, and the SAC development team, changed from 510 to 925. Mandatory dialing of the new area code begins September 12.

1) New Capabilities / Added Commands READDB gets data from an Oracle database. Note that this capability is only available on the Sun/Solaris version of SAC2000. To get READDB working, the executable sac2000.57.oracle.Z, must be downloaded from the ftp site and uncompressed (see the README file which came with the distribution). The user also needs access to a compatibly formatted Oracle database. For details on how the command works, use HELP READDB. A note to the Oracle database manager: The READDB command will work with a database that strictly conforms to the CSS 3.0 data format. Alternatively, if an EVID column is added (and populated) in the WFDISC table, READDB will work, and have increased performance for some queries.

2) Bug Fixes Fixed bug in p2 which prevented fileid from plotting properly if it was set to a header variable.

Fixed bug in central plotting routine which was plotting data points off the right side of the plot.

The DC abbreviation had been overloaded, referring to the DELETECHANNEL command in the main part of SAC2000 and to the DELTACHECK command in the Seismogram Stacking Subprocess. DC no longer refers to the DELTACHECK command. Users wishing to use the DELTACHECK command in SSS should spell out the whole command name.

Fixed a bug in SPECTROGRAM which was producing errors in header fields.

Fixed a bug in 3c which didn't update SAC data in MATLAB when the engine was left running between calls.

— v0.56a —

1) The sac I/O library routines have been rewritten and are included in the SAC library called sac.a. sac.a is available on the SAC2000 ftp site. Now there is one set

of I/O routines which can be called from either C or FORTRAN programs. See help input_output, help blackboard, and help appendix.

— v0.56 and prior —

1) New Capabilities

READCSS does a more comprehensive job of reading CSS 3.0 flat files. see help readcss

READCSS can handle the e1 compressed format.

SAC has a MATLAB interface, allowing users with MATLAB to use its capabilities within SAC. See help mat.

3C is a new three-component processing capability in SAC2000. It allows the user to interactively estimate a signals polarization and wave type (P or S). It can also be run in batch mode. See help 3c

Users who don't have MATLAB, can still use SAC.

The external function capability, which allows users to design their own sac commands, has been enhanced and we have included an example external function called flipxy with the distribution. This function allows the users to transpose XY data, and provides an example to the user of how to write and install an external command into SAC.

WIENER now checks to make sure that the noise window is within the data window.

2) Added Commands DELETECHANNEL (DC) removes a file from SAC's memory. see help deletechannel

FILENUMBER (FN) allows the plots to display the file number of each file. This can be used to determine the number of a file to be deleted with DELETECHANNEL. see help filename.

DELETESTACK performs the same function as DELETECHANNEL, but it can be called from within the Signal Stacking Subprocess (SSS).

PICKAUTHOR and PICKPHASE are used to modify the behavior READCSS uses in reading picks from the .arrival file (CSS 3.0 flat files). see help readcss, help pickauthor, and help pickphase

3) Added Options BB option for TRAVELTIME command saves a time on the black board. see help traveltime

If the DISTANCEWINDOW command is called with two real numbers, it will behave as though the FIXED option is specified.

The utility SGFPLOT can take the CONSTRAIN option which constrains the aspect ratio of the plot to 3/4 (Thanks to Arthur Snoke).

4) Bug Fixes A bug in TRAVELTIME prevented traveltime curves in ascii files from plotting correctly in portrait mode. This has been corrected. see help traveltime

SYNCHRONIZE was synchronizing files properly only if they had the same date. Files with different dates were having the begintimes all set to zero. This has been corrected. For the user who wants the begin times set to zero, the BEGIN option has been added to this command. see help SYNCHRONIZE

A bug prevented users who had changed the color table to return to the default color table (table 17). This has been corrected. see help loadctable

The PLOT1 command would not utilize the relative option if XLIM was set. This has been corrected.

The CUT command now cuts precisely where the user requests.

Recent developments in and enhancements to SAC (v55 and prior)

A number of new features, recent developments or improvements are briefly summarized below. For the most up-to-date information on specific commands consult the on-line documentation. Additional information can also be found on our web page at <http://www-ep.es.llnl.gov/tvp/sac.html>

One of our recent developments is an interface to matlab that will allow you to run matlab scripts from within SAC2000 on data in memory. Our matlab interface currently provides direct access to sac waveform data and all relevant header fields. Development of this interface is still in progress. In our next release, the header will be broken into well defined structures for easier access and we are also considering adding access to SAC2000's blackboard. For more information on this feature type "help mat" inside SAC2000.

Another useful feature that many of our long-time users may not be

aware of is our interface to the Generic Mapping Tools (GMT) codes which allows the user to make quick maps of stations, and events, where the event symbols can be scaled by magnitude or residual.

Users can also develop their own sac commands based on standalone C or FORTRAN programs. These commands run within SAC2000 sharing waveform and header data in SAC2000's internal memory. For more information type help "external_interface"

Our spectrogram/sonogram commands are also a popular feature. These commands produce properly registered high resolution color spectrograms and binary sonograms,

*** The following is a very brief summary of some *** of the recent developments/improvements. *** Commands are in all caps. ***

— Processing enhancements: MAP or GMAP: High Quality Maps using a command line interface to GMT. Arraymap can be used to plot array configurations. external_interface: Make your own sac commands using fortran or c subroutines. LOAD: Loads your external commands. evalresp option in TRANSFER: Transfer has been modified to incorporate evalresp type responses generated by rdseed.

TRAVELTIME: Calculates travetimes based on the iasp91 model. PLOTRECORD-SECTION: new version with a default portrait mode that can also plot iasp91 travel times on record sections and automatically displays travel time picks. It is also possible to zoom in and out and it is no longer necessary to specify a timewindow with this command. ADDSTACK, CHANGESTACK, DELETESTACK, LISTSTACK: begin and end time can now be specified.

SPECTROGRAM: High resolution spectrograms with properly registered seismogram and images. SONOGRAM: High resolution sonograms and binary sonograms. IMAGE: image plots of 3-D data LOADCTABLE (lct): a selection of 17 colortables to use with 3-D images (e.g., spectrograms, fk, etc...)

BBFK and BEAM: Use lat and lon in headers or user7, user8, user9, to compute relative station locations. BBFK: Wavenumber and azimuth at maximum power are written to blackboard variables CORRELATE: output timing made consistent with conventional applications. CONVOLVE: con-

volves a master signal into a number of traces. Timing is now consistent with conventional applications. FUNCGEN: A new function consisting of a string of impulses. LINEFIT: Fit a straight line to data WIENER: filters data using filter design based on a window of noise.

— I/O READCSS: read CSS2.8 and CSS3.0 data into sac. Recent enhancements include a station selection option, (e.g., readcss my.wfdisc station AAA). Shift on or off to set the origin time to reference time (default is on) Scale on or off to scale the data by the calib constant (default is off) Added GSE3.0 format. WRITECSS: write CSS3.0 data from sac. WRITE: Write to xdr format for transformation to linux version. kstcmp option uses to kstnm and kcmpnm to define file name. The kstcmp option is most useful with css data that has multiple waveforms in a single wfdisc. READ: Read from xdr format. (see write).

— Documentation HELP: on-line help contains all the sac documentation. Type help for instructions. Web page: <http://www-ep.es.lnl.gov/tvp/sac.html>

— Convenience and efficiency related enhancements: UNIX: Any unix command except rm can be specified at the command line. HISTORY: view commands or reissue with unix style !"command number" (e.g., !2 to rerun the second command issued in the current sac session).

sacnfiles (an automatically generated blackboard variable): The number of files in memory is now listed as a blackboard variable. CHNHDR: selectively change individual file headers. READ: can now handle path names with wildcards. SETMACRO: Added a new option to add more macro directories and increase maximum number of macro directories to 100. LISTHEADER: You can now see all the header variables with the inclusive option. Even ones that aren't defined. FILENUMBER ON—OFF: When this command is on the filenames are displayed next to the traces. DELETECHANNEL (dc): combined with filename, it allows the user to remove unwanted channels from memory. PLOTPK: a new option to turn the bell on or off: e.g., ppk bell off. (default is on).

— Version 10.6d (6/17/91)—

- (1) Fixed bug with illegal device type causing segmentation fault.
- (2) Fixed bug with continuous prompt when cntrl-d is typed at the com-

mand line.

(3) Fixed problem of no error msg when system command fails due to not enough memory.

(4) Fixed a bug with READ command where WILD and DIR are specified.

— Version 10.6c (6/17/91) —

(1) Fix to OpenWindows cursor problem. Changed the cursor to a cross hair and aligned the hotspot with the vertical and horizontal lines drawn in PPK.

(2) Fix to monochrome monitor problem.

(3) Change to the STRETCH command. Now uses a built-in design routine to generate filter coefficients instead of reading them from a file.

(4) Fix to the READ command to improve use of wild cards and directory path names.

(5) Fix to default graphics device selection. Default is no longer TERMINAL. You will be prompted for the device if none is specified either with the BEGINDEVICE command or through the SACGRAPHICSDEVICE environmental variable.

(6) Fix to the graphics state save and restore in the PLOT1 command.

— Version 10.6b (2/25/91) —

(1) Faster Graphics: plots in buffered mode

— Version 10.6a (1/1/91) —

(1) New commands: MAP - show layout of stations in an array. BBFK - compute the broad band frequency wave number and display the plot. FILTERDESIGN - Graphically displays info about a filter type and parameters. READSDD — WRITESDD - Read and write SDD data files.

(2) Modifications to existing commands: Added ASPECT ON—OFF option to PLOTXY command. Added TRAPEZOIDAL—RECTANGULAR options to INTEGRATE command.

— Version 10.5c (7/3/90) —

(1) Fixed bug involving logarithmic interpolation and fixed plot limits. (2) Decreased length of some contouring subroutine names.

— — Version 10.5b (4/30/90) — — (1) Added ZCOLOR,

ZLABELS, and ZTICKS to contouring package. You now have complete linestyle, label, directional tick mark, and color control of contour plots. (2) Replaced the MINIMUM and MAXIMUM options in ZLEVELS with RANGE. (3) Added LNN and REFTEK instrument responses to TRANSFER. (4) Developed a document revision set for distribution. —————
 ————— — Version 10.5a (3/22/90) ————— (1) Added SPECTROGRAM command. Converts data in memory to a spectrogram. (2) Added CONTOUR command including the ability to set contour levels (ZLEVELS) and linestyles (ZLINES.) All linestyles (not just solid and dotted) have been added since the Seismic Seminar. (3) Added GRAYSCALE command to display data as a grayscale image. Capabilities include scaling, zooming, and windowing. Uses the Utah Toolkit for image display. (4) Added options to the SGF command to control final plot size. (5) Doubled size of data storage array to 2,000,000 words. (6) Fixed bug involving hardware text and sgf. (7) Fixed bug in setting global macro location.

If you have technical questions about this page, contact: peterg@llnl.gov
 – Peter Goldstein

This page maintained by: peterg@llnl.gov – Peter Goldstein
 LLNL Disclaimer 3/8/05 UCRL-MA-112836

6 FAQ

Frequently Asked Questions

1) Sometimes when I run SAC2000 and try to plot some data, the graphics doesn't display. All I get is an empty graphics window.

– Sometimes programs like netscape take hold of the color table etc ... and don't allow SAC2000 to plot. Turn off netscape or other applications that hog the color table. Alternatively, on many systems Netscape can be started up with the -install option; this will let SAC2000 plot, but it will also make some funny colors appear on your screen as you go in and out of Netscape.

2) How do I get a shell variable into SAC2000?

– You can create a temporary ascii file with a command like `setbb variable-name variable` and run it as a macro.

3) How can I reformat a variable with too many zero's? e.g., `getbb x x = 2.0000000e+00`

– You can use the inline functions to select desired portions of the variable you want. See the string manipulation functions `substring`, `before`, and `after`.

For example, `eval as f to x 2 * 1 getbb x = 2.0000000e+00 setbb y (substr 1 3 %x) getbb x = 2.0000000e+00 y = 2.0`

4) Is there a way to convert SEG-2 files (real SEG-2 files, not SEG-Y) to sac data files?

– There is a code called `seg2sac`. It lives at the Passcal facility at L-DGO.

5) Is there any way when producing a record section to have SSS line the waveforms up on a particular phase arrival? For example, if I use `ppk` to pick the first arrival on each waveform and write that to the header, can SSS line the traces up on the arrival?

– Use the `changestack` command as follows: `changestack file# del &file#,pick_header_variable`
For example, `changestack 1 del &t1,t1` would shift the first waveform by the `t1` header value.

6) SAC2000 is supposed to run UNIX commands, but it seems to ignore aliases.

– SAC2000 runs the `korne` shell when executing unix commands, not the `bourne` shell which we usually use. Unfortunately, it is not aware of our usual aliases.

7) SAC2000 is supposed to run UNIX commands, why doesn't `rm` work?

– The `rm` command was disabled because of concerns from some users that they might accidentally type `rm *` instead of `r m *` or some other SAC2000 command. You can use `/bin/rm` filenames if you like.

8) I get an error message "Number 4003" with no explanation.

– Error 4003 is produced under the following conditions: - SAC2000 is trying to read an integer from an ASCII text file or convert a string of numeric characters to an integer. AND - The numeric string contains more than nine characters (in other words, the resultant integer would be greater than 1000000000 (1e+9)).

9) How do I get SAC2000 to read binary data?

– The only binary files SAC2000 reads are in sac format. If you plan to read a file frequently it might be worth converting it to ascii, reading it with readalpha and writing it out as a sac file with the write command.

10) I need SAC2000 for a specific platform. OR I need the sac.a library for a specific platform.

– We are happy to make SAC2000 and/or the sac.a library available to our collaborators. If you are not a collaborator, and would like to be one, please print out the collaborative agreement form from our web site <http://ww-ep.es.lnl.gov/tvp/sac.html> , sign it, and fax it to Peter Goldstein at (925) 423-4077.

The latest versions of SAC2000 are available for Solaris, SunOS, SGI, and Linux. A Alpha version may be available someday.

Fortran libraries are available for Solaris, SunOS, and AIX. A C library is available for Solaris, SunOS, SGI, and Linux.

11) How do I read an ascii file into SAC2000?

- If you have an ascii file in column format

e.g., a file called “myfile”

1.0 3.0 2.0 6.0 3.0 9.0 ... 10.0 30.0

you could use the readalpha command in sac as follows:

ra content xy myfile

- If you have a file in the sac alphanumeric format you would use convert to convert to sac binary format

e.g., convert from alpha myfile

12) After using the cut command to change the lengths of waveforms, how do I get them to begin at a specific time?

- You can use the changehdr (ch) command. Here are two examples:

ch b 0 ch allt 0

13) Why don't my flow control statements work in my c-shell?

- Flow control in sac was only designed to work in macros. When you use flow control interactively or in a script, SAC may not be able to find the end of the flow control loop when its done processing statements in that loop.

For example, if you use an if statement interactively, and the test is false, SAC will not know where to go because there is no endif statement yet. With a macro, it knows to keep looking through the macro file.

14) My Linux version of SAC2000 cannot read sac files written by the UNIX version.

- There is a special option in read and write called xdr. It allows you to transfer unix files to linux. Try the following:

- a) read the data of interest on a machine on a unix platform and write it to a new file such as file.xdr. b) copy this new file to the/ linux machine and read with the xdr option.

- e.g., On unix `sac; read file` `sac; write xdr file.xdr`
on linux `sac; read xdr file.xdr` `sac; write file`

15) I cannot compile the independent programs contained in the utils and demo directories.

- The main utility program you should need is sgftops. This should compile with any c compiler.

- The routines written in FORTRAN are more challenging. One option is to download the FORTRAN library and link to that. Otherwise the fix is platform dependent. Some FORTRAN compilers require calls to functions in C libraries to append an underscore (_) to the function name, and others don't. SunOs and Solaris both require the underscore. For this reason our FORTRAN utilities include the underscore. SGI and Linux do not require the underscore, and for that reason, our FORTRAN utilities don't compile on those platforms yet. We intend to write a wrapper, someday, that will make these utilities portable. We don't know when it will happen.

16) There is no sac/lib/ directory when unpacking the sac2000.linux.tar file.

- The sac.a library has recently been made available at our ftp site for those who have printed out a copy of our collaborative agreement (available on the web at <http://ww-ep.es.llnl.gov/tvp/sac.html>), signed it, and faxed it to Peter Goldstein at (925) 423-4077.

17) When I convert alphanumeric files produced on a VMS system through the command CONVERT, in binary SAC files on a SUN Solaris 3.5 system,

for some data files the program puts up the message WARNING: number 4003 I haven't found in the manual any description of this warning message, could you help me ?

- You should be able to 1) use convert on the vax to generate an alphanumeric sac file 2) copy this alphanumeric file to your sun 3) use convert to read the alphanumeric file on your sun.

The 4003 warning message indicates that SAC2000 tried to read a very large integer (> 1000000000) from ascii, and failed. You may use a text editor to reduce the size of large integers if you feel you can do this in a way that preserves your data.

18) In ppk, when I type 'l' it gives the response to three sig figs. Is there a way to get more sig figs/

- If you make a pick (even just a temporary one) and then use the listhdr command it will give more sig figs, but the time will be relative to kzdate and kztime.

19) The ppk beeps all the time and is driving me nuts. Is there some way to turn off the bell?

- Yes, there is now a BELL OFF option on the ppk command.

20) I can't write files to a remote disk on a Solaris machine.

- We have found that this is a nfs problem and it is fixable with patches.

Solaris 2.5 Patch-ID# 103477-07 Keywords: rpcmod tlimod nfs lock client csh freemem client server stale Synopsis: SunOS 5.5: rpcmod, tlimod and nfs patch Date: Feb/05/97 No reboot is necessary

Solaris 2.5.1 Patch-ID# 103600-13 Keywords: nfs nocto fsync RPC tlimod rpcmod NFS lock clnt_cots CLNT_CALL NLM Synopsis: SunOS 5.5.1: nfs, tlimod and rpcmod patch Date: Apr/28/97

Topic: SunOS 5.5.1: nfs, tlimod and rpcmod patch

NOTE: TO GET THE COMPLETE FIX FOR 4032974, ONE NEEDS TO INSTALL THE FOLLOWING PATCHES: 103640-08 (or higher) Kernel Patch 103934-04 (or higher) kernel/drv/isp patch 104735-01 (or higher) platform/sun4m/kernel/drv/sx patch (for sun4m machines only) 104736-01 (or higher) usr/bin/csh patch *** FAILURE TO INSTALL ALL THESE PATCHES WILL *** CAUSE THE SYSTEM TO HANG AFTER 248

DAYS. ***

103640-08 requires rebooting. The others do not.

21) SAC won't start up properly for me. I get a message that says something like: "Problems opening Command List: /usr/stuff/sac/aux/clstd"

- You may need to set an environmental variable called SAC_AUX. Do this with the setenv command. Set it to the absolute path name of the aux directory that came with sac.

22) The sac.a library is not linking. There are a number of undefined symbols, most of which start with 'X'.

- The new sac library requires the -lX11 flag to appear in the link/compile line.

23) The EVALUATE (or EVAL) command produces floating point values. Is there a way for it to produce integer values?

- Yes, use the AS option, eg.: eval as i to inumber (4 * 5)

24) I have some extra integer information I'd like to store in the header. Could I store it in the NXSIZE or NYSIZE header variables? They don't seem to get used much.

- Maybe, NXSIZE and NYSIZE are used for spectral data, xyz data, and bbfk. If you will be using contouring, bbfk, xyz, or spectral functionality of any sort, you should avoid using NXSIZE and NYSIZE. However, if you know that you won't be using any of this functionality, you can use the command `ch nxsize "integer_value"` from within SAC, or `setnhv('NXSIZE',nx,nerr)` from your own program, where nx is the value you want to write to nxsize

Do it at your own risk.

25) When I use `wsac1()` to over-write a SAC file, I get a segmentation fault.

- If your code has the file name hardwired into the `wsac1` call, try changing it. Instead of: `wsac1('myfile.xyz', ...)` try writing: `fn = 'myfile.xyz'`
`wsac1(fn , ...)`

26) I'm having problems linking with the `sacio.a` library. What could cause these difficulties?

- If your object files (*.o) are out of date it could cause the problem. It may help to rebuild your whole program from the ground up.

- In some cases, the sacio.a library requires linking to the math library. Try adding “-lm” to the link line if it is not already there.

27) Does the FFT in SAC depend on the length of the data window or the sample interval?

- No, The fast fourier transform in SAC approximates the continuous time fourier transform. As such, spectral amplitudes estimated with SAC are independent of the length of the data window or the sampling interval. In other words, two unequal length windows that contain the same signal will have the same fourier spectral amplitudes. However, the sample interval of the spectra are inversly proportional to the length of the time series transformed.

28) Why is it that picks sometimes appear and/or disappear in my files, seemingly without cause?

- Data loaded into SAC is stored in parallel in two different data buffers. There is a SAC formatted data structure, and a CSS formatted data structure. During processing, the data is transfered between the buffers without the user’s knowledge. Because the CSS arrival structure dynamically grows and shrinks as needed, and the SAC time markers T0 through T9 (aka Tn) do not, a conflict arrises when more than 10 picks are present in the CSS data buffer. This was initially solve by putting in place a preferences file to direct the flow of arrivals from the CSS buffer to the SAC buffer. The preferences file is highly structured and sometimes missed important picks which did not fit into the structure. This was addressed in version 0.59 by allowing the user greater flexibility with the behavior of arrivals. This added flexibility, when used without complete understanding, can make picks appear and disappear. To understand how the picks are transfered from the CSS data buffer to the SAC data buffer, use `HELP PICKPREFS`, `HELP READCSS`, `HELP PICKAUTHOR`, and `HELP PICKPHASE`.

7 Availability

SAC was developed at Lawrence Livermore National Laboratory and is copyrighted by the University of California.

SAC is currently licensed to IRIS by LLNL (Lawrence Livermore National Laboratory) through a non-exclusive license agreement. If you are a member of IRIS you can obtain SAC directly from us. Please visit our SAC software pages.

Non-IRIS members should contact LLNL's Industrial Partnerships and Commercialization office at:

Industrial Partnerships And Commercialization Attention: End User License Lawrence Livermore National Laboratory P.O. Box 808 L-795 7000 East Avenue Livermore, CA 94550 E-Mail: softwarelicensing@lists.llnl.gov Main Phone: (925) 422-6416 Fax: (925) 423-8988

<http://www.llnl.gov/ipac/technology/software/softwaretitles/sac.php>

8 Errors/Messages

0000 SAC OUTPUT MESSAGES Last modified on 100991 0000 STRING/NUMERIC MANIPULATION and CONVERSION ERORS 0002 Converting ascii to float - possible bad format. 0000 FILE I/O SERVICE LEVEL 0100 Operating system error 0101 opening file 0102 creating file 0103 for new file 0104 closing file 0105 destroying file 0106 Formatting error encountered while reading file 0107 File unit is in use: 0108 File does not exist: 0109 File already exists: 0110 Illegal file unit number: 0111 File unit is not in use: 0112 truncating file 0113 Illegal file type for file 0114 reading file 0115 writing file 0116 No available file units. 0117 Illegal hardcopy device: 0118 Can't send to 0119 checking existence of file 0120 No wfdisc file specified 0121 Error encoding XDR output file 0122 Partial updates not allowed for XDR file. 0123 Error decoding XDR input file 0124 Can't change to that directory. Check your permissions. 0125 csspickprefs not formatted properly. 0126 .wfdisc filenames require an explicite '.' on the command line: 0127 No data file specified. 0128 CSS file not version 3.0: 0129 Cannot form a path to that file 0130 XDR and ALPHA options are incompatible 0131 sac/datagen data directory not found. 0200 GRAPHICS SERVICE LEVEL 0201 Illegal graphics device 0202 Current graphics device does not have cursor capability. 0203 Can't create X window. Check DISPLAY environmental. 0300 ARRAY

MANAGER FUNCTION 0301 Out of memory. 0302 Memory manager links clobbered for block starting at: 0400 ENLARGE DECOMPRESSION 0401 Enlarge: input record too small 0402 Enlarge: input record too large 0403 Enlarge: record has too many samples 0404 Enlarge: nmap 0405 Enlarge: ndiffs 0406 Enlarge: unexpected end-of-file 0407 Enlarge: nx, nrecl disagree 0408 Enlarge: too few samples found 0409 Enlarge: Number given in record header: 0410 Enlarge: Number of samples decompressed: 0411 Enlarge: inconsistent last values 0412 Enlarge: Last value of original data: 0413 Enlarge: Last value from decompression: 0800 USER SERVICE LEVEL 0801 File is not evenly spaced: 0802 File is not unevenly spaced: 0803 Data truncated to fit in user space for file 0000 GENERAL SERVICE LEVEL 0901 SAC programming logic error 0902 Can't take logarithm of a non-positive number. 0903 Please answer with a YES or NO. 0904 DISTAZ calcuation failed internal check for entry 0905 Time field must be at least 12 characters long. 0906 Date field must be at least 18 characters long. 0907 Bad time field entry detected: 0908 Bad date field entry detected: 0909 Bad julian date field entry detected: 0910 Maximum array that can be sorted is 0911 ***** 0912 (A,I6,I4,I3,I3,I3,I4) 0913 Interrupt received. 0914 Illegal base name: 0915 Illegal base numbers: 0916 File name too long: 0917 Size of passed array(s) too small. 0918 Can't read or write into the global variable file. 0919 SAC data array is too small to execute this command. 0920 Character list delimiter found in character entry: 0921 Not enough room in character list for character entry: 0922 Text would exceed the maximum available space: 0923 Expected to find option in range t0 - t9; none found. 1000 COMMAND MODULE 1001 Bad command syntax at symbol 1002 Bad value for 1003 Value out of allowed range at symbol 1004 Illegal command. 1005 Illegal subprocess command. 1006 Length of string variable exceeded at symbol 1007 Not enough room for command file 1008 No command file name given. 1009 Too many command file arguments at 1010 Wrong number of command file arguments 1011 Bad command file syntax. 1012 Following option is not currently available: 1013 Obsolete command. Please use 1014 Undefined variable in command: 1015 Too many levels of nesting to execute

macro 1016 Terminating execution of macro 1017 Illegal macro command:
 1018 Exceeded maximum number of nested inline functions: 1019 Incorrect
 nesting of inline functions: 1020 Invalid inline function name: 1021 Correct
 number of arguments for this inline function call is 1022 Illegal arithmetic
 operation in inline function: 1023 All arguments to this inline function must
 be numeric. 1024 This argument in inline function should be an operator:
 1025 This argument in inline function should be numeric: 1026 Maximum
 number of arguments for this inline function call is 1027 Exceed maximum
 number of external commands = 1028 External command does not exist:
 1029 Command line too long. 1030 There is no year 0, 1100 EXECUTIVE
 MODULE 1101 Will terminate production run. 1102 Remainder of com-
 mand file not executed. 1103 No help package is available. 1104 No help
 information is available for 1105 Error reading help information for 1106
 Not a valid SAC command. 1107 Invalid entry in sitechan file 1108 UN-
 USED 1109 UNUSED 1110 No news is good news. 1111 Error executing
 system command, insufficient memory. 1112 Error finding program 1113
 Error starting program 1114 Error ending program 1115 This option is not
 currently implemented: 1116 This function is not available on the 1117 Can't
 evaluate expression because of bad operand value: 1118 Maximum number
 of open transcript files is 1119 Maximum number of active traceable vari-
 ables is 1200 VARS FUNCTION 1201 Could not find VARS variable 1202
 Maximum number of vars sections exceeded: 1203 Could not find VARS
 section 1204 Incorrect data type for VARS variable 1205 Could not delete
 VARS variable 1206 VARS option not currently implemented: 1207 Bad
 data block flag for VARS variable 1208 Disk file is not in VARS format:
 1209 No current vars section has been defined. 1210 Bad input to subrou-
 tine 1211 VARS list already exists: 1300 DATA FILE MODULE 1301 No
 data files read in. 1302 Maximum memory size exceeded. 1303 Overwrite
 flag is not on for file 1304 Illegal operation on data file 1305 Illegal operation
 on time series file 1306 Illegal operation on unevenly spaced file 1307 Ille-
 gal operation on spectral file 1308 Maximum smoothing half width is 1309
 Maximum special header list length is 1310 Illegal data file list number 1311
 No list of filenames to write. 1312 Bad number of files in write file list: 1313

Illegal relative time pick 1314 Data file list can't begin with a number. 1315
 Maximum number of files in data file list is 1316 Can't smooth an unevenly
 spaced data file. 1317 The following file is not a SAC data file: 1318 Header
 in disk file is out of date: 1319 Bad data found in card image data file header.
 1320 Available memory too small to read file 1321 Can't cut spectral file
 1322 Undefined starting cut for file 1323 Undefined stop cut for file 1324
 Start cut less than file begin for file 1325 Stop cut greater than file end for
 file 1326 Start cut greater than file end for file 1327 Stop cut less than file
 begin for file 1328 Start cut greater than stop cut for file 1329 Corrected by
 filling with zeros. 1330 Corrected by using file begin. 1331 Corrected by us-
 ing file end. 1332 Fatal error condition. 1333 Unable to read some files 1334
 Can't read or write DS2 formatted data files 1335 Illegal operation—only
 data file headers in memory. 1336 Undefined header field value. 1337 Illegal
 header field name. 1338 Too many data points to perform operation for file
 1339 Too few data points to perform operation for file 1340 data points out-
 side allowed range contained in file 1341 Can't write headers because CUT
 is ON. 1342 Illegal number of files in data file list: 1343 Formatting error
 while reading file 1344 Problem writing GSE file 1350 Could not find re-
 quested header entry 1351 Not enough room in header for new header entry
 1352 Can not delete header entry 1353 Output variable too short for header
 entry 1354 No end-of-header found. 1355 Incorrect data type for header
 entry 1356 Can't cut unevenly spaced data file 1357 Decoding formatted
 alphanumeric data card. 1358 Maximum number of free format entries ex-
 ceeded: 1359 Maximum number of alphanumeric data channels exceeded:
 1360 Illegal character in alphanumeric content descriptor: 1361 Can only
 have one X channel per file. 1362 Must have at least one Y channel per
 file. 1363 Illegal data file list name: 1364 No data file list specifier (name
 or number) given. 1365 Illegal enumerated header field value: 1366 This
 command requires that data in memory be of type XYZ: 1377 Unable to
 adjust the time in the SDD header 1378 Illegal operation on XYZ data 1379
 No SORT parameters given 1380 Too many SORT parameters: 1381 Not a
 valid SORT parameter: 1382 ALL and COMMIT options both set, ignoring
 COMMIT option. 1383 SORT failed 1384 ASCEND and DESCEND options

go after the related header in the command line 1385 No worksets in memory. 1386 Could not get Workset name. 1387 No file name specified. 1388 Reference time not equal to zero: KZDATE and KZTIME may be adjusted by subsequent commands so that the reference time is zero. Reference time is 1389 NVHDR, NPTS, NWFID, NORID, and NEVID cannot be changed with CHNHDR 1390 KSTNM and KCMPNM cannot be undefined 1391 Cannot CUTIM: would result in too many files in memory. 1392 Cannot CUTIM: would exceed length of filename list (use shorter filenames) 1393 Cannot Write Table of file: 1394 Unexpected option on PICKPREFS; expecting ON, OFF, or blank. 1400 SEISMGR SAC INTERFACE 1401 Data may be corrupt. Proceed with caution. 1402 Data may have been removed. Proceed with caution. 1403 Cannot CUTIM: would result in too many files in memory. 1404 Cannot CUTIM: would exceed length of filename list (use shorter filenames) 1405 Cannot CUTIM: illegal cut point information 1406 Data has been corrupted, re-read or regenerate new data. 1500 GRAPHICS ACTION MODULE 1501 Floor used 1502 Bad cursor position. Please retry. 1503 Invalid character. Please retry. 1504 Probable discrepancy in reference date fields in headers. 1505 Must specify at least two data file list names or numbers. 1600 SPECTRAL ANALYSIS MODULE 1601 File and filter sampling intervals not equal for 1602 Inadequate memory to perform FIR filter using DFT. 1603 Inadequate memory to perform FIR filter. 1604 Following file now in amplitude-phase format: 1605 Following file now in real-imaginary format: 1606 Maximum allowable DFT is 1607 DC level after DFT is 1608 Bad Wiener filter noise window for file 1609 Numerical instability in Wiener filter for file 1610 Unwrap failed at data point for file 1611 Corner frequency greater than Nyquist for file 1612 Window length exceeds maximum: 1613 Minimum size of data file for Hilbert transform is 1614 Numerical instability in Wiener; will retry with epsilon = 1615 Noise window outside of data window 1616 Noise window larger than data window 1617 Noise window partially outside of data window 1618 Order = 0 in HQR 1619 HQR failed, too many iterations 1620 Gain out of range, Filterdesign failed for Whitening coefficients for file: 1700 UNARY OPERATIONS MODUDE 1701 Can't divide by zero. 1702 Non-positive values found in file

1800 BINARY OPERATIONS MODULE 1801 Header field mismatch: 1802
 Time overlap: 1803 No binary data files read in. 1804 Illegal binary data
 file list number: 1805 Time gap (zeros added): 1900 EVENT ANALYSIS
 MODULE 1901 Can't open HYPO pick file 1902 Can't open card image
 pick file 1903 Can't close previous card image pick file. 1904 All global card
 image pick files are in use. 1905 Need an integer. Retry. 1906 Need an
 integer in the range 0 to 4. Retry. 1907 HYPO line already written. 1908
 HYPO pick file not open. 1909 Can't compute waveform. 1910 No valid
 pick found for the following file(s): 1911 Can't estimate back azimuth be-
 cause of 2000 SIGNAL CORRECTION MODULE 2001 Command requires
 an even number of data files. 2002 Following files are not an orthogonal
 pair: 2003 Following files are not both horizontals: 2004 Insufficient header
 information for rotation: 2005 Points outside file's time window set to zero
 = 2006 Gains must be monotonically decreasing. 2007 Data clipped for file
 2008 Requested begin time is less than files begin time. Output truncated.
 2009 Requested end time is greater than files end time. Output truncated.
 2010 Number of points in pair of files are not equal: 2011 Cannot read fil-
 ter coefficient file: 2100 INSTRUMENT CORRECTION MODULE 2101
 Need free period and magnification for ELMAG. 2102 Need number of zeros
 for EYEOMG. 2103 Need number of zeros, free period, scale and damping
 factors for GENL. 2104 Need an instrument sub-type for 2105 Unknown
 instrument sub-type for 2106 Need free period and damping factor for LLL
 sub-type BB. 2107 Need free period, damping factor, and corner frequency
 for PORT. 2108 Maximum number of poles exceeded in POLEZERO file:
 2109 Maximum number of zeros exceeded in POLEZERO file: 2110 Illegal
 option in POLEZERO file: 2111 Taper frequency limits are invalid. No taper
 applied. 2112 Incorrect value for free period or magnification for ELMAG.
 2113 Need free period, damping, corner, gain, and highpass for REFTEK.
 2114 No response information for this channel in response file. 2115 No re-
 sponse file found in database 2116 Not a recognized response file type. 2117
 SUBTYPE and FNAME options not compatible with DBASE, filenames
 ignored. 2118 No transfer function applied. 2119 The SCALE option is
 not required if the TRANSFER command is to be used on data. 2120 In-

terpolation Failed: adjacent frequencies indistinguishable. Freq: 2121 NDC transfer had an OS error. 2122 NDC transfer had an application error. 2123 NDC transfer had a SQL error. 2124 NDC transfer had an unknown error. 2200 GRAPHICS DEVICE MODULE 2201 First three elements in color table entry must be numeric: 2202 Size of passed color table arrays are too large. 2203 Bad values in color table arrays found and corrected: 2204 opening font file: 2205 reading font file: 2300 GRAPHICS DEVICE 1 2301 No TERM environmental variable set. 2400 GRAPHICS DEVICE 2 2401 Can't find an unused SAC Graphics File. 2402 Can't PRINT on ENDFRAME if SGF device is not on. 2403 Ignoring PRINT option in the middle of a frame. 2404 SPECTROGRAM, SONOGRAM, and IMAGE only PRINT if SGF is the only graphics device running 2405 Cannot PRINT: no SGF files produced. 2500 GRAPHICS DEVICE 3 2600 GRAPHICS DEVICE 4 2700 CONDITIONAL EXECUTION MODULE 2701 Syntax error in DO statement 2702 Do loop list exceeds maximum number of characters = 2703 Can't evaluate logical expression: 2704 Reading macro file 2705 Searching macro file for 2800 NEURAL NETWORK MODULE 2801 All data files must have the same number of data points. 2900 XYZ (3-D) DATA PROCESSING MODULE 2901 No xyz data in memory. 2902 Zoomed input too large to display. Maximum dimension is 3000 CONTOURING MODULE 3001 Exceeded maximum number of contouring levels: 5000 SPECTRAL ESTIMATION SUBPROCESS. 5001 Spectral Estimation Subprocess. 5002 Only one file can be processed by SPE at a time. 5003 No correlation function calculated. 5004 No spectral estimate calculated. 5005 Error within Dave Harris's subroutine package. 5006 A single evenly spaced data file is not in memory. 5007 Confidence limits option not currently implemented. 5000 SIGNAL STACKING SUBPROCESS. 5101 Signal Stacking Subprocess. 5102 No files in stack file list. 5103 No time window defined. 5104 No distance defined for file(s) 5105 Time window mismatch: 5106 File name not in file list: 5107 File number not in file list: 5108 Maximum length of stack file list exceeded: 5109 Sampling intervals are not equal. 5110 Illegal velocity model number: 5111 Error in calculating velocity model values. 5112 Insufficient input for velocity model calculation. 5113 No valid stack sum exists.

5120 Cannot use both model and file. Continuing with model. 5121 Data file
 expected, none found. 5122 Distance out of range of data; blackboard vari-
 able not set: 5123 No blackboard variable name given, no variable set. 5124
 TAUP and MODEL options are incompatable in TRAVELTIME. Input file
 required 5125 No phases found 5200 FIR FILTER DESIGN SUBPROCESS
 5300 FK module 5301 Station and event latitudes and longitudes must be
 set for this command. 5302 Unable to determine offsets, type HELP BEAM
 5303 OFFSET set to REF, but no reference data, use REFERENCE op-
 tion 5304 OFFSET set to USER, but some files missing USER7 or USER8
 5305 OFFSET set to STATION, but some files missing STLA or STLO 5306
 OFFSET set to EVENT, but some files missing EVLA or EVLO 5307 Il-
 league setting for OFFSET option 5308 Number of files must be between 3
 and MXLENS 6000 DATA SET MODULE 6001 Can't find file in data-set
 file index. 6002 No more data-sets available. 6003 Max number of files in
 data-set memory. No more room. 6004 Invalid data set name. Must be a
 character string. 6005 Another data-set already exists by this name 6006
 Maximum allowed number of current data-set exceeded. 6008 Bad syntax
 in command. 7000 MAP PROGRAM 7001 Invalid map projection number
 7002 Invalid map window number 7003 Invalid map pole 7004 Invalid map
 viewport 7005 Map projection and window type are inconsistent 7006 Illegal
 window size 7007 Location could not be transformed 7008 Exceeded maxi-
 mum size of a data array: 7009 Illegal option found on card: 7010 Unable to
 open ZONESDATA file. 7011 Unable to open gctp messages file tmp.????
 7100 CYLINDRICAL PROJECTION: PARALLELS selects only latitude of
 true scale. 7101 EQUIDISTANT CYLINDRICAL PROJECTION: PPOLE
 command parameters ignored. 7102 TRANSVERSE MERCATOR: PAR-
 ALLELS command ignored in this projection. 7103 ILLEGAL VALUE:
 Scale at meridian greater than 1.0 7104 CONIC PROJECTION: PPOLE
 command parameters ignored. 7105 OBLIQUE MERCATOR: Illegal pro-
 jection pole. 7200 Invalid latitude for center of UTM map. Can't get ZONE.
 8001 SETMAT takes only one parameter. 8002 Cannot link to MATLAB
 shared object: 8003 Cannot link to a MATLAB function: 8004 Cannot start
 MATLAB 8100 ORACLE DATABASE CONNECTION 8101 command or

option not operational; requires Oracle database version of SAC2000

9 Developers

10 Copyright

11 Privacy and Leagal Notice